Department of Computer Science and Engineering

CPE 598(A &B)
&
CPE 597C

Thesis: Image Morphing

Prof. Ausif Mahmood

Submitted By : Pallavi Mahajan
(ID# : 664711)

Acknowledgement

I am grateful to Prof. Ausif Mahmood for his guidance and immense support during my Master's Thesis. He has been a constant source of knowledge and I would like to take this as an opportunity to thank Prof. Ausif Mahmood for giving me knowledge and encouragement throughout my Master's program. I would also like to thank the Dean of School of Engineering, Dr. Tarek Sobh for his kind support throughout my MS.

Table of Content

INTRODUCTION

Recently there has been a great deal of interest in *morphing* techniques for producing smooth transitions between images. These techniques combine 2D interpolations of shape and color to create dramatic special effects. Part of the appeal of morphing is that the images produced can appear strikingly lifelike and visually convincing. Despite being computed by 2D image transformations, effective morphs can suggest a natural transformation between objects in the 3D world. The fact that realistic 3D shape transformations can arise from 2D image morphs is rather surprising, but extremely useful, in that 3D shape modeling can be avoided.

Morphing is probably most noticeably used to produce incredible special effects in the entertainment industry. It is often used in movies such as *Terminator* and *The Abyss*, in commercials, and in music videos such as Michael Jackson's *Black or White*. Morphing is also used in the gaming industry to add engaging animation to video games and computer games. However, morphing techniques are not limited only to entertainment purposes. Morphing is a powerful tool that can enhance many multimedia projects such as presentations, education, electronic book illustrations, and computer-based training. We discuss what morphing is, different morphing techniques and examples of morphing software packages available for multimedia developers to use when creating multimedia projects.

Significance of Topic

Animation techniques are constantly increasing in quality and creativity. Because consumers demand better quality special effects and effects that will captivate and engage themselves grows profoundly, industries must strive to appease these audiences. Multimedia users and entertainment seekers are no longer satisfied with simple animation, they desire fancier transitions and animation to amaze them and keep them interested in the product. The special effect known as morphing has been one way to satisfy consumers' need to be entertained, impressed, and amazed by the final product.

Discussion of Topic

What is Morphing?

The word morph derives from the word metamorphosis meaning to change shape, appearance or form. According to Vaughn (112) morphing is defined as “an animation technique that allows you to dynamically blend two still images, creating a sequence of in-between pictures that, when played in QuickTime, metamorphoses the first image into the second.” Yongyue Zhang gives a detailed explanation of the process of morphing: Morphing is achieved by coupling image warping with color interpolation. As the morphing proceeds, the source image is gradually distorted and is faded out, while the target image is faded in. So, the early images in the sequence are much like the first image. The middle image of the sequence is the average of the first image distorted halfway towards the second one and the second image distorted halfway back towards the first one. The last images in the sequence are similar to the second one. Then, the whole process consists of warping two images so that they have the same "shape" and then cross dissolving the resulting images. Another term that warrants being defined is ‘warping’ because it is frequently used when discussing the process of metamorphosis. A warp is a two-dimensional geometric transformation and generates a distorted image when it is applied to an image (Thalmann). Warping is similar to morphing, except that no fade occurs and only one image is distorted (Cybulski and Valentine). According to Claypoole, et al. there are two ways to warp an object. The first method is forward mapping in which each pixel in the source image is mapped to an appropriate place in the destination image. The second is reverse mapping, which goes through each pixel in the destination image and samples an appropriate source image pixel.

Morphing involves the image processing techniques of cross-fading and warping to morph one image into a completely different image. Morphing software allows a step by step transformation from one image into another.

We divide the task into two categories, morphing algorithms and automatic feature detection algorithms. The most difficult one is wrapping of one image into another image. It is the starching and pulling of the images, that makes morphing effects so realistic. The actual morphing of the images can be accomplished either by using morph points or morph lines. Morph points are the markers that you set up on the start image and the end image. The morphing program then uses these markers to calculate how the initial image should bend/wrap to match the shape of the final image. The second method is lines instead of individual points. Both methods produce very realistic morphing image.

In my thesis I am accomplishing the morphing by using morph points. One of the most time consuming tasks in morphing is selecting the points or lines in the initial and final image so that the metamorphosis is smooth and natural. An algorithm to automatically select the morphing markers would save a great deal of time and money.

The process of morphing is not all that complex, but like rendering , it takes a lot of processing horsepower to accomplish. To morph between two images, we have to first digitize the " before" and "after" pictures into the computer. Next, we have to choose series of sequences points on the before picture. Then we have to mark the same number of pints on the after picture so that each point directly corresponds to a point placed on the before picture. Decisions is to be taken about how many intermediate pictures is to be make and computer does the rest once the process is complete, you have a series of images that create a illusion of motion. The points can be moved, edited, deleted or added whatever it takes to create final realistic series of images.

WARPING

There are two ways to warp an image. They are:

1) FORWARD WARPING:

In this method each pixel in the source image is mapped to an appropriate place in the destination image. Thus, some pixels in the destination image may not be mapped. We need interpolation to determine these pixels values. This mapping was used in our point-morphing algorithm.

POINT WARPING

This method of image warping is based on a forward technique. Where each pixel from the input image is mapped in a new position in the output image. Since not every output pixel will be specified, we must use an interloping function to complete the output image. We specify several control points which will map exactly to given location in the output image. The neighboring pixels will move somewhat less than the control point, with the amount of movement specified by a weighting function consisting of two separate components, both dependent on the distance from the pixel to each control point in the image. The first component of the weighing function Gaussian function which is unity at the control point and decays to zero as you move away from the control point. The idea is to have pixels far away from a control point be unaffected by the movement of that point. The problem with this scheme is each pixel is affected by the weighing function of each control point in the image. So even though the weighing function at a control may be one, that point will still be affected by the movement of every other control point in the image, and won't move all the way to its specified location.

In order to overcome this effect, I designed the second component of the weighing function, which depends on the relative distance from a pixel to each other control point. The distance to the nearest control point is used as a referenced, and the contribution of each control point is reduced by the factor depending on the distance to the nearest control point divided by the distance to that control points.

2) REVERSE MAPPING

This method goes trough each pixel in the destination image and samples an appropriate source image pixel. Thus all destinations image pixels

are mapped to some source image pixels. This mapping has been used in the line morphing method.

Morphing with Lines

Morphing between two images 10 and 11 is done as follows. Lines are defined on the two images 10 and 11. The mapping between the lines is specified depending on the number of intermediate frames required, a set of interpolated lines are obtained. An intermediate frames is obtained by doing three things The lines in the first image 10 are warped to the lines corresponding to the intermediate image. The lines in the second image 11 are warped to the lines corresponding to the intermediate image. The two warped images are now combined proportionality depending on how close the frame is with respect to the initial and final frames. Since the images have been warped to the same shape before cross-dissolving the intermediate image looks good.

In either case, the problem is to determine the way in which the pixels in one image should be mapped to the pixels in the other image. So we need to specify how each pixel moves between the two images. Specifying the mapping for a few important pixels could do this. The motion of the other pixels could be obtained by appropriately corresponding the information specified for the control pixels. These sets of control pixels can be specified as points mapping to points.

Morphing Techniques

Image morphing techniques can be classified into two categories such as mesh-based and feature-based methods in terms of their ways for specifying features. In mesh-based methods, the features on an image are specified by a nonuniform mesh. Feature-based methods specify the features with a set of points or line segments. (Thalmann) One way of achieving the morphing effect is to transform one image into another by creating a cross-dissolve between them. According to Claypoole et.al., in this method, the color of each pixel is interpolated over time from the first image value to the corresponding second image value. However, this is not very effective in portraying an actual metamorphosis and the metamorphosis between faces does not look good if the two faces do not have about the same shape. This method also tends to wash away the features on the images (Thalmann).

A second way to achieve morphing is feature interpolation, which is performed by combining warps with the color interpolation. The features of two images and their correspondences are specified by an animator with a set of points or line segments. Then, warps are computed to distort the images so that the features have intermediate positions and shapes. The color interpolation between the distorted images finally gives an inbetween image (Thalmann).

In morphing the most difficult task is the warping of one image into another image (Claypoole et.al.). It is the stretching and pulling of the images that makes the morphing effect so realistic. The actual morphing of the image can be accomplished either by using morph points or morph lines. Morph points are the markers that you set up on the start image and the end image. The morphing program then uses these markers to calculate how the initial image should bend/warp to match the shape of the final image. The second method uses lines (edges) instead of individual points. Both methods produce very realistic morphing effects. One of the most time consuming tasks in morphing is selecting the points or lines in the initial and final image so that the metamorphosis is smooth and natural. There are several useful tips to remember when morphing objects. The first is to choose carefully those pictures to morph (Morphing Software). For example, if you wish to morph two animals, it is best to use pictures that have the same general size and outline.

If one picture of the animal is a close up of the head then the other picture should also be a close up of the head to obtain successful results. A second tip is to carefully select the background (Morphing Software). If a single color background is used, the morphing effect focuses on the object. Ideally, it is best to use the same background.

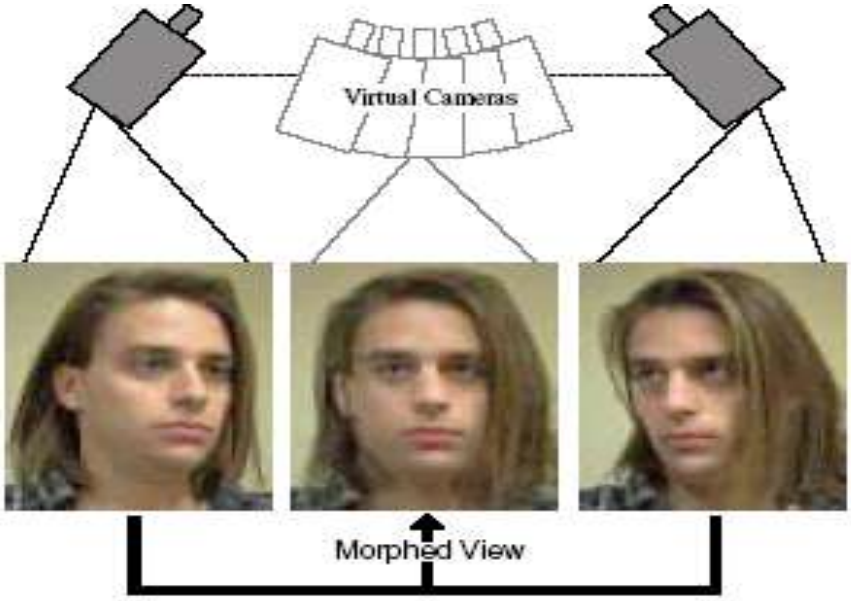


Figure 1: View morphing between two images of an object taken from two different viewpoints produces the illusion of physically moving a virtual camera.

Part of the problem is that existing image morphing methods do not account for changes in viewpoint or object pose. As a result, simple 3D transformations (e.g., translations, rotations) become surprisingly difficult to convey convincingly using existing methods. We describe a simple extension called *view morphing* that allows current image morphing methods to easily synthesize changes in viewpoint and other 3D effects. When morphing between different views of an object or scene, the technique produces new views of the same scene, ensuring a realistic image transition. The effect can be described by what you would see if you physically moved the object (or the camera) between its configurations in the two images and filmed the transition, as shown in Fig. 1. More generally, the approach can synthesize 3D *projective transformations* of objects, a class including 3D rotations, translations, shears, and tapering deformations, by operating entirely on images (no 3D shape information is required). Because view morphing employs existing image morphing techniques as an intermediate step, it may also be used to interpolate between different views of *different* 3D objects, combining image morphing's capacity for dramatic shape transformations with view morphing's ability to achieve changes in viewpoint.

The result is a simultaneous interpolation of shape, color, and pose, giving rise to image transitions that appear strikingly 3D.

View morphing works by prewarping two images, computing a morph (image warp and cross-dissolve) between the prewarped images, and then postwarping each in-between image produced by the morph. The prewarping step is performed automatically, while the postwarping procedure may be interactively controlled by means of a small number of user-specified control points. Any of several image morphing techniques, for instance [15, 1, 8],may be used to compute the intermediate image interpolation.

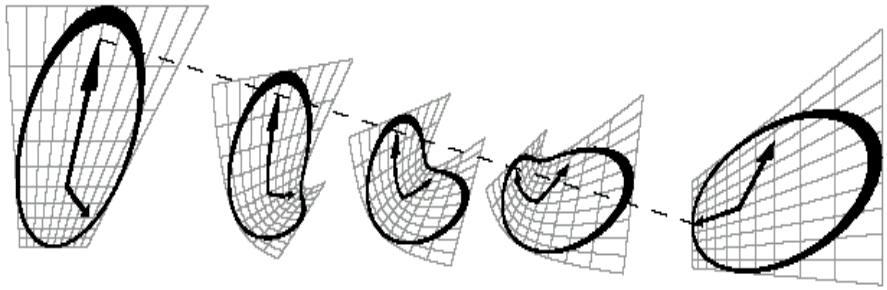


Figure : A Shape-Distorting Morph.

Linearly interpolating two perspective views of a clock (far left and far right) causes a geometric bending effect in the in-between images. The dashed line shows the linear path of one feature during the course of the transformation. This example is indicative of the types of distortions that can arise with image morphing techniques. This not require knowledge of 3D shape, thereby allowing virtual manipulations of unknown objects or scenes given only as drawings or photographs. In terms of its ability to achieve changes in viewpoint, view morphing is related to previous view-based techniques such as viewsynthesis [3, 7, 11, 12] and mosaics [10, 2, 14, 6]. However, this paper focuses on creating natural *transitions b*etween images rather than on synthesizing arbitrary views of an object or scene. This distinction has a number of important consequences. First, in computing the transition between two perspective views, we are free to choose a natural camera path. By choosing this path along the line connecting the two optical centers, we show that the formulation and implementation is greatly simplified. second, our approach is general in that it can be used to compute transitions between any two images, thereby encompassing both rigid and nonrigid transformations.

In contrast, previous view-based techniques have focused on rigid scenes. Finally, view morphing takes advantage of existing image morphing techniques,

already in widespread use, for part of the computation. Existing image morphing tools may be easily extended to produce viewmorphs by adding the image prewarping and postwarping steps.

IMAGE MORPHING

Image morphing, or *metamorphosis*, is a popular class of techniques for producing transitions between images. There are a variety of morphing methods in the literature, all based on interpolating the positions and colors of pixels in two images. At present, there appears to be no universal criterion for evaluating the quality or realism of a morph, let alone of a morphing method. A natural question to ask, however, is does the method preserve 3D shape. That is, does a morph between two different views of an object produce new views of the same object? Our investigation indicates that unless special care is taken, morphing between images of similar 3D shapes often results in shapes that are mathematically quite different, leading to surprisingly complex and unnatural image transitions. These observations motivate *view morphing*, introduced in the next section, which preserves 3D shape under interpolation. We write vectors and matrices in bold face and scalars in roman. Scene and image quantities are written in capitals and lowercase respectively. When possible, we also write corresponding image and scene quantities using the same letter. Images, I, and 3D shapes or scenes, S, are expressed as point sets. For example, an image point

(x; y) = p 2
is the projection of a scene point
(X; Y;Z) = P 2 S.

A morph is determined from two images I0 and I1 and maps
(C0 : I0) I1 and C1 : I1) I0

specifying a complete correspondence between points in the two images. Two maps are required because the correspondence may not be one-to-one. In practice, C0 and C1 are partially specified by having the user provide a sparse set of matching features or regions in the two images. The remaining correspondences are determined automatically by interpolation [15, 1, 8]. A warp function for each image is computed from the correspondence maps, usually based on linear interpolation:

W0(p0; s) = (1 _ s)p0 + sC0(p0) (1)
W1(p1; s) = (1 _ s)C1(p1) + sp1 (2)

W0 and W1 give the displacement of each point p0 2 I0 and p1 2 I1 as a function of s 2 [0; 1]. The in-between images Is are computed by warping the two original images and averaging the pixel colors of the warped images. Existing morphing methods vary principally in how the correspondence maps are computed. In addition, some techniques allow finer control over interpolation rates and methods. For instance, Beier et al. [1] suggested two different methods of interpolating line features, using linear interpolation of endpoints, per Eqs. (1) and (2), or of position and angle. In this paper, the term *image morphing* refers

specifically to methods that use linear interpolation to compute feature positions in in-between images, including [15, 1, 8].

To illustrate the potentially severe 3D distortions incurred by image morphing, it is useful to consider interpolating between two different views of a planar shape. Any two such images are related by a 2D projective mapping of the form:

$$H(x; y) = \left(\frac{ax + by + cgx + hy + i}{dx + ey + f}; \frac{gx + hy + i}{}\right)$$

Projective mappings are not preserved under 2D linear interpolation since the sum of two such expressions is in general a ratio of quadratics and therefore not a projective mapping. Consequently, morphing is a *shape-distorting* transformation, as in-between images may not correspond to new views of the same shape. A particularly disturbing effect of image morphing is its tendency to bend straight lines, yielding quite unintuitive image transitions. Fig. 2 shows a Dali-esque morph between two views of a clock in which it appears to bend in half and then straighten out again during the course of the transition. The in-between shapes were computed by linearly interpolating points in the two views that correspond to the same point on the clock.

Changes in Visibility

So far, I have described how to correct for distortions in image morphs by manipulating the projection equations. Eq. (3), however, does not model the effects that changes in *visibility* have on image content. From the standpoint of morphing, changes in visibility result in two types of conditions: *folds* and *holes*. A fold occurs in an in-between image Is when a visible surface becomes folded. Prewarping is possible if the images are first cropped to exclude the epipoles occluded in Is. In this situation, multiple pixels of I0 map to the same point in Is, causing an ambiguity. The opposite case, of an occluded surface suddenly becoming visible, gives rise to a hole; a region of Is having no correspondence in I0. Folds can be resolved using Z-buffer techniques [3], provided depth information is available. In the absence of 3D shape information, we use point *disparity* instead. The disparity of corresponding points p0 and p1 in two parallel views is defined to be the difference of their x-coordinates. For parallel views, point disparity is inversely proportional to depth so that Z-buffer techniques may be directly applied, with inverse disparity substituted for depth. Because our technique makes images parallel prior to interpolation, this simple strategy suffices in general. Furthermore, since the interpolation is computed one scanline

at a time, Z-buffering may be performed at the scanline level, thereby avoiding the large memory requirements commonly associated with Z-buffering algorithms. An alternative method using a Painter'smethod instead of Z-buffering is presented in [10].

Unlike folds, holes cannot always be eliminated using image information alone. Chen and Williams [3] suggested different methods for filling holes, using a designated background color, interpolation with neighboring pixels, or additional images for better surface coverage. The neighborhood interpolation approach is prevalent in existing image morphing methods and was used implicitly in our experiments.

Producing the Morph

Producing a shape-preserving morph between two images requires choosing a sequence of projection matrices

s = [H s j
_ H s
C s]

There are many ways to specify this transformation. A natural one is to interpolate the orientations of the image planes by a single axis rotation. If the image plane normals are denoted by 3D unit vectors N0 and N1, the axis D and angle of rotation _ are given by

D = N0 _N1
_ = cos_1(N0 _ N1)

Alternatively, if the orientations are expressed using quaternions, the interpolation is computed by spherical linear interpolation [13]. In either case, camera parameters such as focal length and aspect ratio should be interpolated separately.

Controlling the Morph

To fully determine a view morph, Hs must be provided for each inbetween image. Rather than specifying the 3 _ 3 matrix explicitly, it is convenient to provide Hs indirectly by establishing constraints on the in-between images. A simple yet powerful way of doing this is to interactively specify the paths of four image points through the entire morph transition. These control points can represent the positions of four point features, the endpoints of two lines, or the bounding quadrilateral of an arbitrary image region2. Fig. 6 illustrates the

process: first, four control points bounding a quadrilateral region of 0:5 are selected, determining corresponding quadrilaterals in I0 and I1. Second, the control points are interactively moved to their desired positions in I0:5, implicitly specifying the postwarp transformation and thus determining the entire image I0:5. The postwarps of other in-between images are then determined by interpolating the control points. The positions of the control points in Is and ^Is specify a linear system of equations whose solution yields Hs [15]. The four curves traced out by the control points may also be manually edited for finer control of the interpolation parameters. The use of image control points bears resemblance to the view synthesis work of Laveau and Faugeras [7], who used five pairs of corresponding image points to specify projection parameters. However, in their case, the points represented the projection of a new image plane and optical center and were specified only in the original 2Care should be taken to ensure that no three of the control points are colinear in any image. In our approach, the control points are specified in the *in between* image(s), providing more direct control over image appearance.

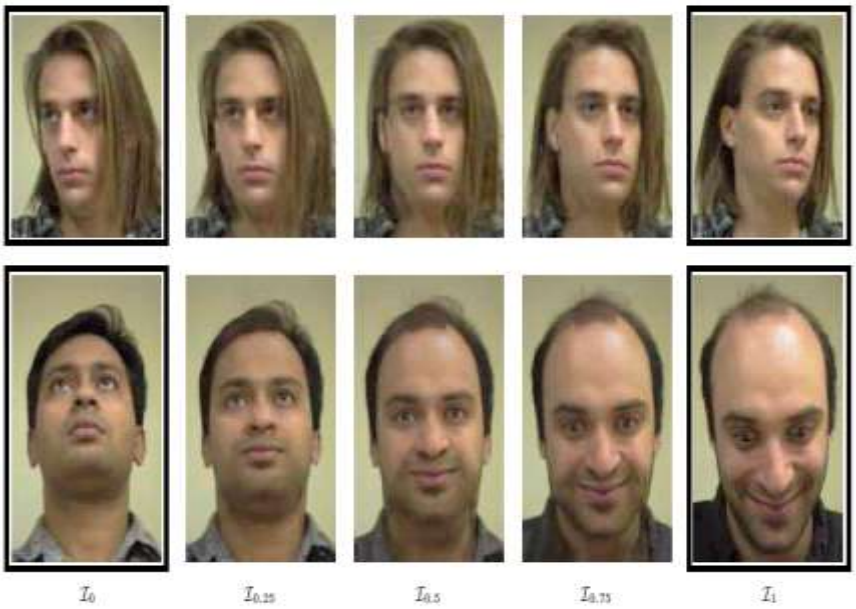


Figure: Facial View Morphs. Top: morph between two views of the same person. Bottom: morph between views of two different people.

In each case, view morphing captures the change in facial pose between original images I0 and I1, conveying a natural 3D rotation.

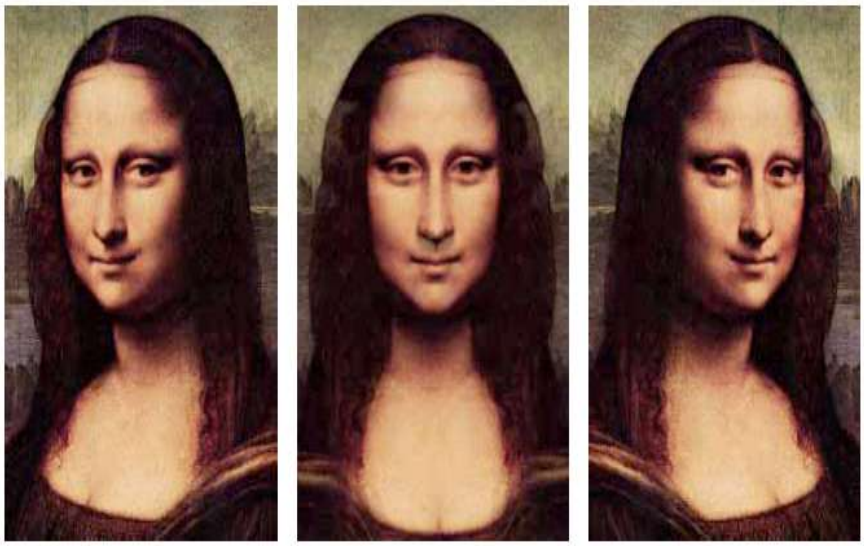


Figure : Mona Lisa View Morph. Morphed view (center) is halfway between original image (left) and it's reflection (right).

Recent Advances in Image Morphing

Image morphing has been the subject of much attention in recent years. It has proven to be a powerful visual effects tool in film and television, depicting the fluid transformation of one digital image into another. This paper reviews the growth of this field and describes recent advances in image morphing in terms of three areas: feature specification, warp generation methods, and transition control. These areas relate to the ease of use and quality of results. I will describe the role of radial basis functions, thin plate splines, energy minimization, and multilevel free-form deformations in advancing the state-of-the-art in image morphing. Recent work on a generalized framework for morphing among multiple images will be described.

Image metamorphosis has proven to be a powerful visual effects tool. There are now many breathtaking examples in film and television depicting the fluid transformation of one digital image into another. This process, commonly known as *morphing*, is realized by coupling image warping with color interpolation. Image warping applies 2D geometric transformations on the images to retain geometric alignment between their features, while color interpolation blends their color. Image metamorphosis between two images begins with an animator establishing their correspondence with pairs of feature primitives, e.g., mesh nodes, line segments, curves, or points. Each primitive specifies an image feature, or landmark. The feature correspondence is then used to compute mapping functions that define the spatial relationship between all points in both images. Since mapping functions are central to warping, I shall refer to them as warp. Presented at: Computer Graphics International ’96, Pohang, Korea. June 1996 functions in this paper. They will be used to interpolate the positions of the features across the morph sequence. Once both images have been warped into alignment for intermediate feature positions, ordinary color interpolation (i.e., cross-dissolve) is performed to generate inbetween images. Feature specification is the most tedious aspect of morphing. Although the choice of allowable primitives may vary, all morphing approaches require careful attention to the precise placement of primitives. Given feature correspondence constraints between both images, a warp function over the whole image plane must be derived. This process, which we refer to as warp generation, is essentially an interpolation problem. Another interesting problem in image morphing is transition control. If transition rates are allowed to vary locally across inbetween images, more interesting animations are possible.

The explosive growth of image morphing is due to the compelling and aesthetically pleasing effects possible through warping and color blending. The extent to which artists and animators can effectively use morphing tools is directly tied to solutions to the following three problems: feature specification, warp generation, and transition control. Together, they influence the ease and

effectiveness in generating high-quality metamorphosis sequences. This paper describes recent advances in image morphing in terms of their role in addressing these three problems. Comparisons are given between various morphing techniques, including those based on mesh warping [14], field morphing [2], radial basis functions [1], thin plate splines [9, 6], energy minimization [7], and multilevel free-form deformations [8]. A tradeoff exists between the complexity of feature specification and warp generation. As feature specification becomes more convenient, warp generation becomes more formidable. The recent introduction of spline curves to feature specification raises a challenge to the warp generation process, making it the most critical component of morphing. It influences the smoothness of the transformation and dominates the computational cost of the morphing process.

Figure : Cross-dissolve

Morphing Algorithms

Before the development of morphing, image transitions were generally achieved through the use of cross-dissolves, e.g., linear interpolation to fade from one image to another.

Fig. depicts this process applied over five frames. The result is poor, owing to the double-expos re effect apparent in misaligned regions. This problem is particularly apparent in the middle frame, where both input images contribute equally to the output. Morphing achieves a fluid transformation by incorporating warping to maintain geometric alignment throughout the cross-dissolve process. In this section we review several morphing algorithms, including those based on mesh warping, field morphing, radial basis functions, thin plate splines, energy minimization, and multilevel free-form deformations. This review is intended to motivate the discussion of progress in feature specification, warp generation, and transition control.

Mesh Warping

Mesh warping was pioneered at Industrial Light& Magic (ILM) by Douglas Smythe for use in the movie *Willow* in 1988. It has been successfully used in many subsequent motion pictures. To illustrate the 2-pass mesh warping algorithm, consider the image sequence shown in Fig. 2. The five frames in the middle row represent a metamorphosis (or morph) between the two faces at both ends of the row. For simplicity, the meshes are constrained to have frozen borders. All intermediate frames in the morph sequence are the product of a 4-step process:

for each frame f do
    linearly interpolate mesh M, between M_S and M_T
    warp I_S to I_1, using meshes M_S and M
    warp I_T to I_2, using meshes M_T and M
    linearly interpolate image I_f, between I_1 and I_2
end

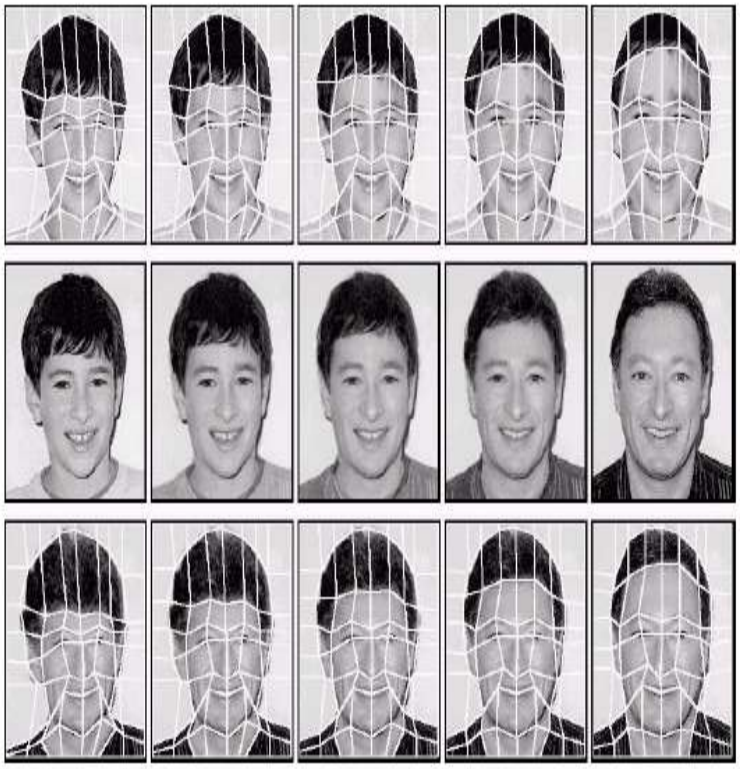


Figure : Mesh warping

Field Morphing

While meshes appear to be a convenient manner of specifying pairs of feature points, they are, however, sometimes cumbersome to use. The field morphing algorithm developed by Beier and Neely [2] at Pacific Data Images grew out of the desire to simplify the user interface to handle correspondence by means of line pairs. A pair of corresponding lines in the source and target images defines a coordinate mapping between the two images. In addition to the straightforward correspondence provided for all points along the lines, the mapping of points in the vicinity of the line can be determined by their distance from the line. Since multiple line pairs are usually given, the displacement of a point in the source image is actually a weighted sum of the mappings due to each line pair, with the weights attributed to distance and line length. This approach has the benefit of being more expressive than mesh warping. For example, rather than requiring the correspondence points of Fig. 2 to all lie on a mesh, line pairs can be drawn along the mouth, nose, eyes, and cheeks of the source and target images. Therefore only key feature points need be given. Although this approach simplifies the specification of feature correspondence, it complicates warp generation. This is due to the fact that all line pairs must be considered before the mapping of each source point is known. This global algorithm is slower than mesh warping, which uses bicubic interpolation to determine the mapping of all points not lying on the mesh. A more serious difficulty, though, is that unexpected displacements may be generated after the influence of all line pairs are considered at a single point. Additional line pairs must sometimes be supplied to counter the ill-effects of a previous set. In the hands of talented animators, though, the mesh warping and field morphing algorithms have both been used to produce startling visual effects.

The progression of morphing algorithms has been marked by more expressive and less cumbersome tools for feature specification. A significant step beyond meshes was made possible by the specification of line pairs in field morphing. The complications that this brought to warp generation, however, sometimes undermined the usefulness of the approach. For instance, the method sometimes demonstrated undesirable artifacts, referred to as *ghosts*, due to the computed warp function [2]. To counter these problems, the user is required to specify additional line pairs, beyond the minimal set that would otherwise be warranted. All subsequent algorithms, including those based on radial basis functions, thin plate splines, and energy minimization, formulated warp generation as a scattered data interpolation problem and sought to improve the quality (smoothness) of the computed warp function. They do so at relatively high computational cost. The newest approach, based on the MFFD algorithm,

significantly improves matters by accelerating warp generation. The use of snakes further assists the user in reducing the burden of feature specification.

Transition Control

Transition control determines the rate of warping and color blending across the morph sequence. If transition rates differ from part to part in in between images, more interesting animations are possible. Such nonuniform transition functions can dramatically improve the visual content.

Note that the examples shown thus far all used a uniform transition function, whereby the positions of the source features steadily moved to their corresponding target positions at a constant rate.

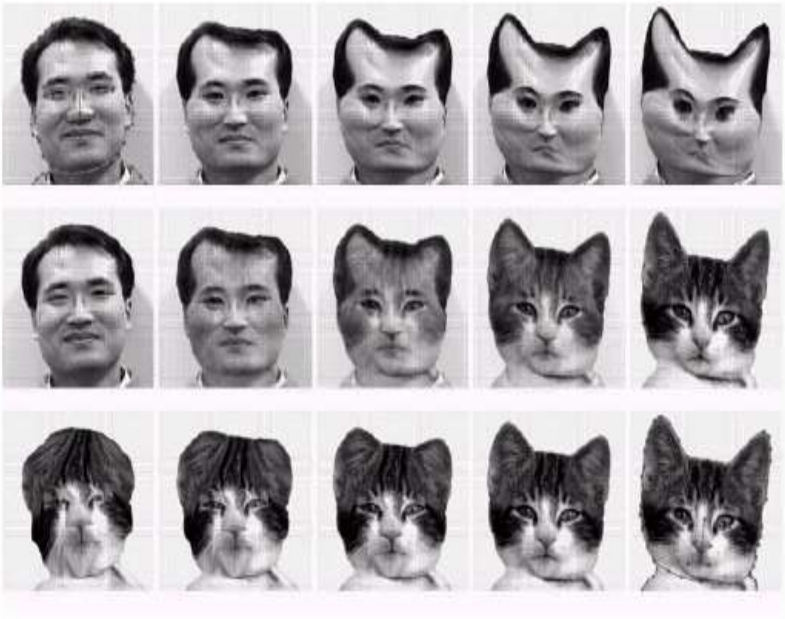


Figure 4. Uniform metamorphosis

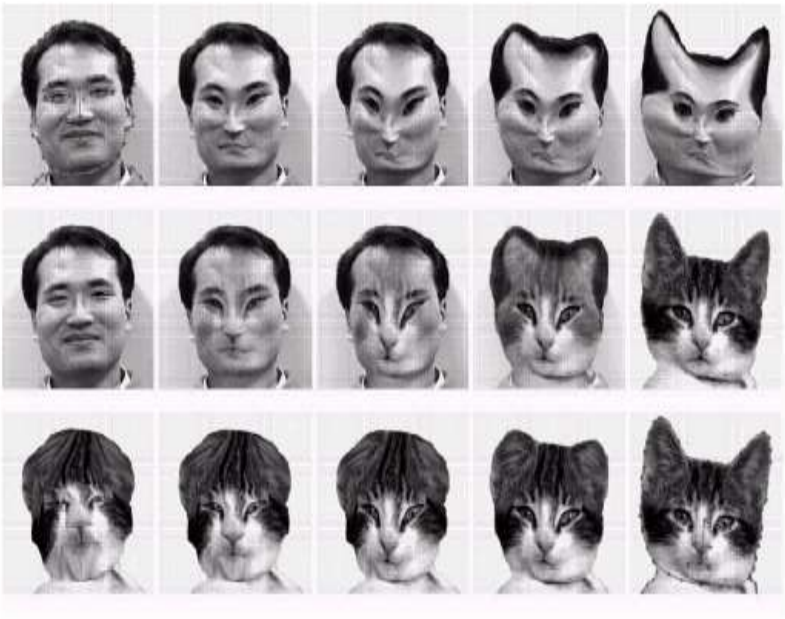


Figure 5. Nonuniform metamorphosis

Figures above show examples of the use of uniform and nonuniform transition functions, respectively. The upper left and lower right images of Fig. are the source and target images, respectively. The features used to define the warp functions are shown overlaid on the two images. The top and bottom rows depict a uniform transition rate applied to the warping of the source and target images, respectively. Notice, for instance, that all points in the source and target images are moving at a uniform rate to their final positions. Those two rows of warped imagery are attenuated by the same transition functions and added together to

yield the middle row of in between images. Note that geometric alignment is maintained among the two sets of warped in between images before color blending merges them into the final morph sequence.

The example in Fig. demonstrates the effects of a non uniform transition function applied to the same source and target images. In this example, a transition function was defined that accelerated the deformation of the nose of the source image, while leaving the shape of the head intact for the first half of the sequence. The deformation of the head begins in the middle of the sequence and continues linearly to the end. The same transition function was used for the bottom row. Notice that this use of non uniform transition functions is responsible for the dramatic improvement in the morph sequence.

Transition control in mesh-based techniques is achieved by assigning a transition curve to each mesh node. This proves tedious when complicated meshes are used to specify the features. It can be mentioned that the transition behavior can be controlled by a B´ezier function defined on the mesh [10].

In the energy minimization method, transition functions are obtained by selecting a set of points on a given image and morphing algorithms generally coupled the feature specifi-cation and transition control primitives, this method permits them to be decoupled. That is, the location of transition control primitives must not necessarily coincide with those of the features. The transition curves determine the transition behavior of the selected points over time. For a given time, transition functions must have the values assigned by the transition curves at the selected points.

The examples in Figs. 4 and 5 were generated using the MFFD-based morphing algorithm. Transition curves can be replaced with procedural transition functions [7, 8]. An example is depicted in Fig. 6, where a linear function varying in the vertical direction is applied to two input images. The result is a convincing transformation in which one input image varies into the other from top to bottom. The traditional formulation for image morphing considers only two input images at a time, i.e., the source and target images. In that case, morphing among multiple images is understood to mean a series of transformations from one image to another. This limits any morphed image to take on the features and colors blended from just two input images. Given the success of morphing using this paradigm, it is reasonable to consider the benefits possible from a blend of more than two images at a time. For instance, consider the generation of a facial image that is to have its eyes, ears, nose, and profile derived from four different input images.

In this case, morphing among multiple images is understood to mean a seamless blend of several images at once.

Despite the explosive growth of morphing in recent years, the subject of morphing among multiple images has been neglected. In ongoing work conducted by the author and his colleagues, a general framework is being developed that extends the traditional image morphing paradigm applied to two images.

Morphing among multiple images is ideally suited for image composition applications where elements are seamlessly blended from two or more images. A composite image is treated as a metamorphosis of selected regions in several input images. The regions seamlessly blend together with respect to geometry and color. In future work, we will determine the extent to which the technique produces high quality composites with considerably less effort than conventional image composition techniques. In this regard, the technique can bring to image composition what image warping has brought to cross-dissolve in deriving morphing: a richer and more sophisticated class of visual effects that are achieved with intuitive and minimal user interaction. Future work in morphing will also address the automation of morphing among limited classes of images and video sequences. Consider a limited, but common, class of images such as facial images. It should be possible to use computer vision techniques to automatically register features between two images. As the examples given in Figs. 2 and 3 demonstrate, facial images require feature primitives to be specified along the eyes, nose, mouth, hair, and profile. Model-based vision should be able to exploit knowledge about the relative position of these features and automatically locate them for feature specification [3]. Currently, this is an active area of research, particularly for compression schemes designed for videoconference applications. The same automation applies to morphing among two video sequences, where time varying features must be tracked. Interested readers may refer to the recent proceedings of the 1995 IEEE International Conference on Image Processing (Washington, D.C) for several papers on facial image processing and motion tracking.

Problem Statement

The emerging trend of graphics technology is imperative that the intricacies of the device that support them be known in addition to the technology used to implement application further. The aim of this project is to study this technology to uncover the problem s faced by the application programmers due to memory and display constraints. The various options available for developing an application in VC++ are to be studied and a suitable option is to be selected for implementation. The phases of the software development life cycle (SDLC) are to be followed for the same.

The implementation of the project involves two images which are first digitized. These digital images then need to be converted to compatible equal frame dimensions and are processed to show the transformations from one to the another step by step or phase by phase to show clearly the mapping from the features of one into the individual feature of the other.

Application Name – Image Morphing

Applications of the thesis

- Animation Movies
- Medical Application Simulations
- Films
- Machine Designing.

System concept development phase

System [or project] concept development actually starts the life cycle when a need to develop or significantly change a system is identified once a business need, based on operation requirement, is identified and documented, the approaches for meeting it are reviewed for feasibility and appropriateness. The need may involve development of a new system of modification of an existing system. Senior official approvals and funding are needed before beginning the planning phase.

System concept development begins when a need is formally identified as requiring study and analysis that may lead to system development activities. To get to this point, the agency must recognize that such a need exists and then articulate that need to a decision-maker.

Planning Phase

A program management plan in development that document the approach to be used and includes the discussion of methods, tools, tasks, resources, project schedules, and user input.

Many of the plan essential to the success of the entire project are create in this phase; the plan created are than reviewed and undated throughout the remaining phases. The plan created within this phase include the QA plan which is reviewed and amplified upon as necessary within the requirement analysis phase, it is then finalized in the design phase; the project management plan created in this phase is reviewed and changed a necessary is requirement analysis design and development phases and will then be finalized in the integration and test phase. The CM created in this phase will be reviewed and changed if necessary in the requirement analysis, design, and the development phase and will be finalized in the integration and phase.

Requirement analysis phase

Functional user requirement are formally defined and delineate the requirement in terms of data, system performance security and maintainability requirement for system. All requirements are defined to a level of details sufficient for systems design to proceed.

The requirement analysis phase will be initiated when project is approved for development in the initial planning review or by management direction. Documentation related to user requirement from the planning phase shall be used as the basic for further user needs analysis and the development of detailed user requirements. The analysis may reveal new insights into the overall information systems requirements, and in such instance, all deliverable should be revised to reflect this analysis.

During the Requirement Analysis phase, the system shall be defined in more detail with required to system inputs, processes, outputs and interfaces (both internal and external). This definition process occurs at the functional level. The system shall be described is terms of the functions to be performed, not in terms of computer programs, files and data streams.

Design Phase

The external physical characteristics of the system are designed during this phase. The operating environment is established, major sub system and their inputs and outputs are defined and processes are allocated to resources. Everything requiring user input or approval must be documented and reviewed by the user. The internal physical characteristics of the system are specified and a detailed design is prepared. Subsystems defined during the external design are used to create a detailed structure of the system. Each subsystem is partitioned into one or more design units or modules. Detailed logic specifications are prepared for each software module.

The objective of the design phase is to transform the detailed, defined requirements into complete, detailed specification for the system to guide the work of the development phase. The decisions made in the phase address, in detail, how the system will meet the defined functional, physical interface, and data requirements. Design phase activities may be conducted in an interactive fashion, producing first a general system design that emphasizes the functional features of the system, then a more detailed system design that expands the general design by providing all the technical detail.

Development Phase

The detailed specifications produced during the design phase are translated into hardware, communication and executable software. Software shall be unit tested, integrated and retested in a systematic manner. Hardware is assembled and tested.

The objective of the Development Phase will be to convert the deliverables of the Design phase a complete information system.

Although much of the activity in the development phase addresses the computer programs that make up the system, this phase also puts in place the hardware, software and communication environment for the system and other important elements of the overall system.

The activities of this phase translate the system design produced in the design phase into a working information system capable of addressing the information system requirements. The development phase contains activities for requirement analysis; design coding, integration, testing and installation and acceptance related to software products. At the end of this phase, the system will be ready for the activities of the integration and test phase.

`

Integration and Test phase

Subsystem integration system, security and user acceptance testing is conducted during the integration and test phase. The user with the quality assurance organization validates that the functional requirements, as defined in the functional requirements documents, are satisfied by the developed or modified system.

Several types of tests will be conducted in this phase. First subsystem integration test shall be executed and evaluated by the developed team to prove that the program components integrated properly into the subsystem and that the subsystems integrate properly into an application. Next the testing team conducts and evaluates system tests to ensure the developed system meets all technical requirements, including performance requirements. Next the testing team and the security program manager conduct security tests to validate that the access and data security requirements are met. Finally, users

participate in acceptance testing to confirm that the developed system meets all user requirements as stated in the problem statements. Acceptance testing shall be done in a simulated “real” user environment with the user using simulated or real target platforms and infrastructures.

Implementation Phase

The system or system modification are installed and made operational in a production environment. The phase is initiated after the system has been tested and accepted by the user. This phase continues until the system is operating in production in accordance with the defined user requirements.

In this phase, the system or system modifications are installed and made operational in a production environment. The phase is initiated after the system has been tested and accepted by the user. Activities in this phase include notification of implementation to end users, execution of the previously defined training plan, data entry or conversion, completion of security certification and accreditation and post implementation evaluation. This phase continues until the system is operating in production in accordance with the defined user requirements.

The new system can fall into three categories, replacement of a manual process, replacement of a legacy system, or upgrade to an existing system. Regardless of the type of system, all aspect of the implementation phase should be followed. This will ensure the smoothest possible transition to the organization’s desired goal.

Operational and Maintenance phase

The system operation is ongoing. The system is monitored for continued performance in accordance with user requirements, and needed system modifications are incorporated. Operations continue as long as the system can be effectively adapted to respond to an organization’s need. When modifications or changes are identified as necessary, the system may reenter the planning phase.

More than half of the cycle costs are attributed to the operations and maintenance of the system. In this phase, it is essential that all facets of operations and maintenance are performed. The system is being used and scrutinized to ensure that it meets the needs initially stated in the planning phase. Problems are detected and new needs arise. This may require modification to existing code to be developed and/or hardware configuration changes. Providing user support is an ongoing activity. New users will require training and other will require training as well. The emphasis of this phase will be to ensure that the users needs are met and the system continues to perform as specified in the operational environment. Additionally, as operations and maintenance personnel monitor the current system they may become aware or better ways to improve the system and therefore make recommendations. Changes will be required to fix problems, possibly add features and make improvements to the system. This phase will continue as long as the system is in use.

SOFTWARE PROCESS MODEL

We have used the Linear Sequential Model, also known as the Waterfall Model because the classic life cycle paradigm has a definite and important place in software engineering works. It is still the most widely used process model used for software engineering tasks. Though armed with some weaknesses, it is better than a haphazard approach to software development. Also the Linear Sequential Model is the oldest and the most trusted one.

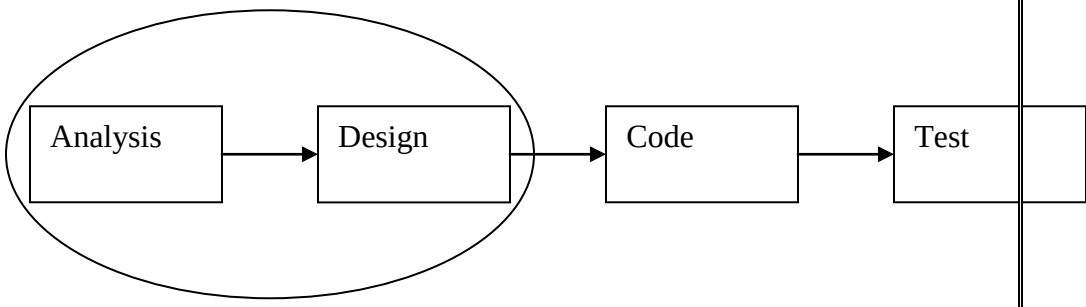


Figure: Linear Sequential Model

- System engineering and analysis encompasses requirements' gathering at the system level with a small amount of top level analysis and design.
- Information engineering encompasses requirements' gathering at a strategic level and business area level.
- In analysis, software system requirements are analysed. The requirements gathering process is intensified and focused specially on software.
- Software design is actually a multi-step process that focuses on four distinct attributes of a program: data structures, software architecture, interface representation and procedural details. The design process translates the requirements into a representation of the software that can be assessed for quality before code generation begins.
- The code generation step performs the task of translating the design into machine readable form.
- Once the code has been generated, program testing begins. The testing process focuses on the logical internals of the software, assuring that all

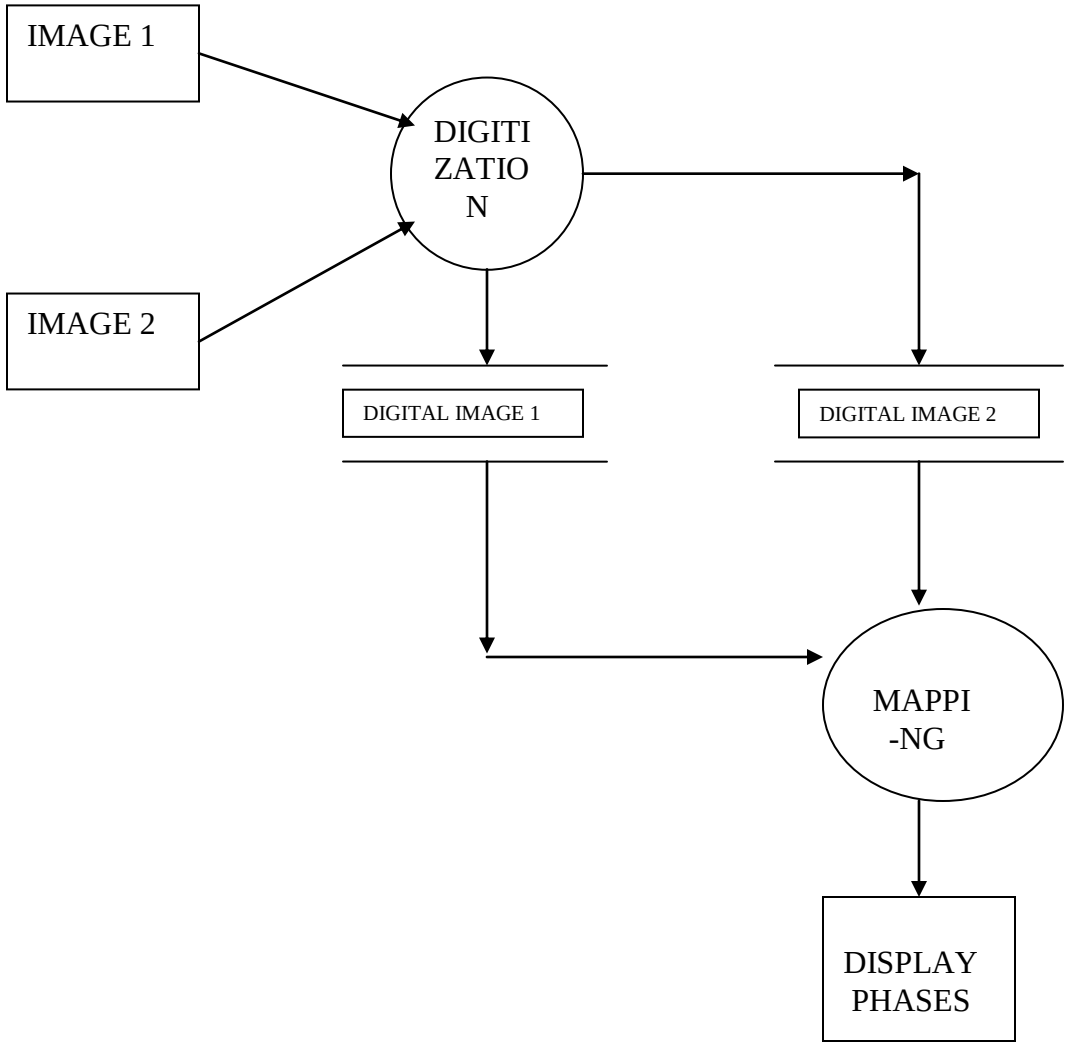
statements have been tested, and all the functional externals - that is, conducting tests to uncover errors and ensure that defined input will produce actual results which agree with required results.

SYSTEM / INFORMATION ENGINEERING

SYSTEM ANALYSIS

- The software should be user friendly and easy to use.
- He image to be morphed can be scanned, taken from a webcam or stored in memory as a jpeg file or an extract from a mpeg file.
- The user is placed on a level of abstraction, wherein the size, resolution and the type of the image(Tag Image File Format) need not be a concern.
- The user can decide on the number of phases through which the morphing and mapping go through to produce the required morphed image.
- The final morphed image is device independent, that is, can be viewed on any display device.
- The project supports 16K colours.

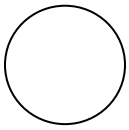
INFORMATION FLOW DIAGRAM



Basic notations of Information flow diagram:

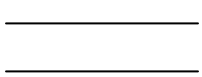
External Entity: A procedure or information that resides outside the bounds of the system. It is represented by:

Process : A transformer of the information(a function) that resides within the bounds of the system to be modeled. It is represented as:

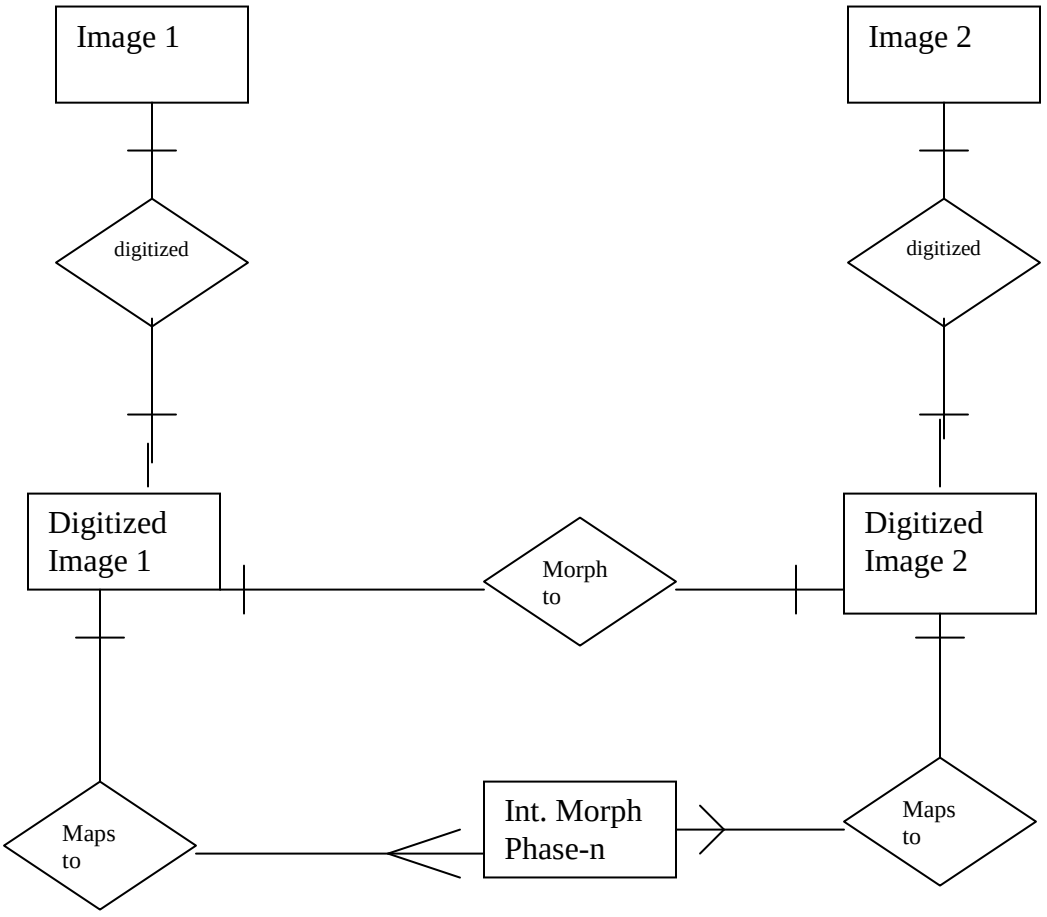


Data Object : A data object, the arrowhead indicates the direction of data flow. It is represented as:

Data Stores : A repository of data that is to be stored for use by one or more processes; Maybe as simple as a buffer or queue or as sophisticated as a relational database. It is represented as:



ENTITY RELATIONSHIP DIAGRAM



Basic Notations of E-R-Diagram

Data Objects: A data object is a representation of almost any composite information that must be understood by software.

Relationships: Data objects are connected to one another by means of relationships that exists between them.

Cardinality: Cardinality is the specification of the number of occurrences of one object that can be related to the number of occurrences of another object. Two objects can be related as- one-to-one, many- to-many and one-to-many.

Hardware and software requirements

The package is to be designed for Windows, to be used on IBM or compatible machines. The only support is the VGA display card and machine which should be Windows 95/98 compatible. The programming language chosen is Visual C++.

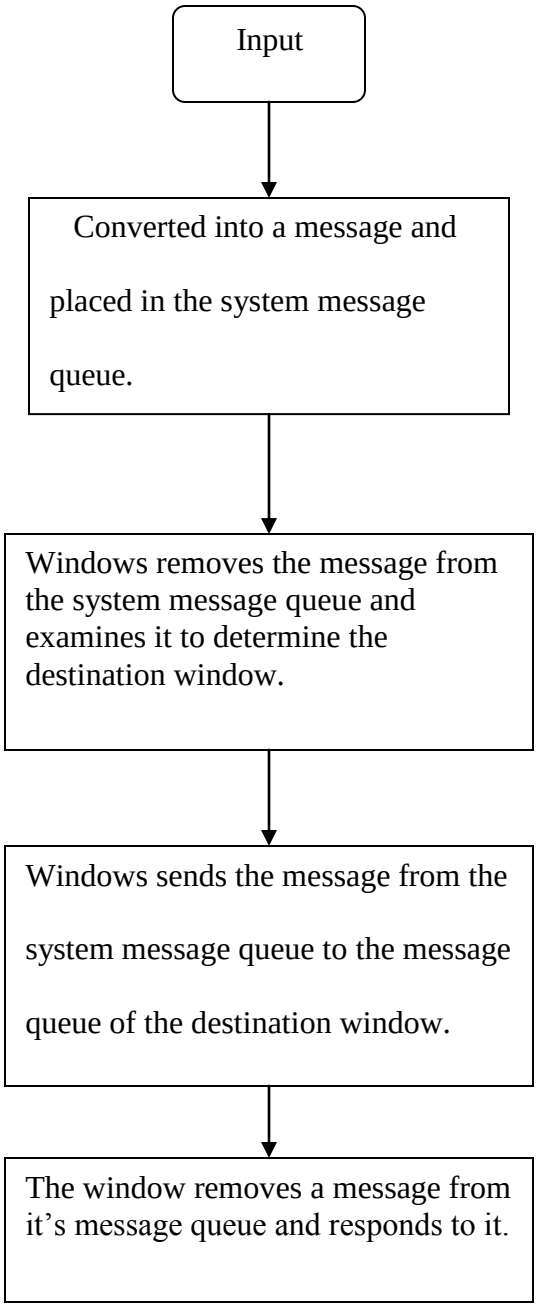
Following are the basic system configuration requirements namely,

- 486 or higher processor.
- At least 16 MB RAM of memory(32MB RAM is ideal).
- Super VGA display system with 256 colours or more.
- Windows 95 or higher version.
- Scanner.
- VC++ software with MSDN support.

PLATFORM SELECTED

The topic is based on platform selected i.e.windows for the thesis IMAGE MORPHING. Windows is special graphics screen based software developed by Microsoft.we have particularly used windows 98,which is an operating system. Windows owes it's name to the fact that it runs each program inside a separate window. A window has a box or a frame on the screen. You can have numerous windows on the screen at a time,each containing it's own program. You can then easily switch between the program without having to close one down and open the next. The question always may arise that why we have used windows? Because it has many advantages over MS-DOS. The switching over from MS-DOS to windows operating system involves graphical user interface from the typical text interface that the MS-DOS offers.

Windows programs deal with the input device like keyboard and mouse that is directed to numerous interface objects such as menus and buttons, which at anytime can call anyone of the functions. This means that it has an advantage on the sequence driven nature of DOS program. The windows program is driven by the events that takes place during the execution of the program.


Input
Converted into a message and placed in the system message queue.
Windows removes the message from the system message queue and examines it to determine the destination window.
Windows sends the message from the system message queue to the message queue of the destination window.
The window removes a message from it's message queue and responds to it.

OVERVIEW OF WINDOWS PROGRAMMING:

Windows program are different than DOS programs. Following are the basic principle under which windows program works.

1) Instead of telling the computer what to do one step at a time, windows programs are structured to wait there until the program receives messages from windows messages are statement like" the user just click a button with the mouse pointer do something!".

2) The windows environment has built in support for all the basic hardware such as video display, memory, mouse, keyboard and printers. Microsoft takes care of worrying about all of the latest hardware freeing you to create application. Programmers spend their time learning and using the 600 windows functions, rather than writing their own code to support multiple printers, video cards etc.

3) Windows moves programs and data around in memory to make a room for other program pieces and data. This movement allows many programs to co-exist in a fairly limited amount of memory, but it all means that the programmer cannot assume that anything will stay put for long. Windows gives you all of the tools you need to deal with movable memory, but it takes little getting used to.

The EXE mode in windows is remarkably small conserving that you have a resizable graphics window icon and many functions built in and full mouse support. The secret to the small size is that window program do not contain ever traction of the program code needed to do all of these operations. The program you create makes use of a larger collections of functions that are part of the windows environment when windows is running on your computer. Every windows program share these working libraries of functions of control of the screen, printers, keyboards, mouse, menus, bitmaps and a long list of other functions.

This collection of working function is maintained in files stored in the system directly on your hard disk. The three primary files are GDI.EXE, USER.EXE and KERNEL.EXE which provides video display and printer functions, mouse, keyboard, sound, communication part and timer support and files and memory management respectively.

1) Windows KERNEL (KERNEL.EXE):

This part of windows 95 provides support for the lower level functions that an application needs to run. For example, every time your application needs memory, it runs to the windows kernel to get it. This component does not deal with either the interface or devices: it interacts only with windows itself.

2) Graphical Device Interface (GDI.EXE):

Every time an application writes to screen, it uses a GDI service. This windows component takes care of fonts, printer services, the display, color management and every other artistic aspect of windows that users can see as they use your application.

3) USER.EXE:

windows is all about just that – windows. It needs a manager to keep a track of all the windows that application creates to display various types of information . However, user only begins their it's using some type of user component function.

LANGUAGE SELECTED

Why use windows ?

There are several reasons why we use windows for developing any application.

1. Device Independent Programs:

Windows lets you write device independent program. This means that while you are writing the application, you do not concern yourself with what type of printers, mouse monitors, keyboard, sound card, CD-ROM, device or other device your user own. Your application will work fine , no matter what hard ware your user have.

2. Pre-Installed Code:

A lot of code is already installed in your user's PC. Ones windows is installed the PC contains much windows selected software. This code exists on your PC(i.e. the developer's PC) and on your user PC. This means that before you even start writing the first line of code yourself, your user has already more than half your program in his or her PC!

3. Standard User Interface:

The user interface mechanism is the same for all windows application. Without reading your applications documentation, user know how to execute your application, they can use the icon on the corner of the application window to minimize or maximize the window, they know the meaning of OK and CANCEL button, they know what about the dialog box is and they understand many other features of your program before you even start writing it.

These are the reasons for using windows for developing a particular application. There are various programming languages for developing a window and application such as Microsoft Access, Visual Basic and Visual C++.

Why VC++?

In VC++ we use C and C++ for writing code. Therefore the meaning of C++ in Visual C++ is understood. What does visual mean? It means you will accomplish many of your programming task by using the keyboard and mouse to visually design and write your applications. You will select controls such as push buttons, radio buttons and scroll bars with the mouse, drag them to your application windows as you build it(this is called design time).

In other words you will be able to see what your application will look like before you execute it. This is a great advantage because it will save your considerable time and you can see your application with completing linking it and you can change your mind about placement and size of edit boxes, pull buttons and other objects simply by using the mouse.

Components Of The Systems:

The fundamental parts of VC++ provided as part of IDE are the editor, the compiler, the linker and the libraries(these are the basic tools of C++).

THE EDITOR:

It provides an interactive environment for creating and editing C++ source code as well as usual facilities such as cut and paste. Additionally the editor automatically recognizes fundamental words in the C++ language and assigns a color to them according to what they are. This not only helps to make your code more readable, but also provides a clear indicator of when you make errors in keying such words.

THE COMPILER:

As usual the compiler converts your source code into machine language. The output from compiler is known as object code and stored in files called object files, with extension object.

THE LINKER:

The linker combines the various modules generated by the compiler from source code files, adds required code modules from program libraries supplied as a part of C++ and welds everything into an executable whole. The linker can also detect and report errors. For example- if part of your program is missing or a non existent library.

THE LIBRARIES:

It supports and extends C++ language by providing routine to carry out operations that are not part of the language. There are several kinds of libraries provided by VC++. The first kind contains routines that aren't platform- specific . There are standard libraries as in C++.

The other libraries provided by VC++ are collectively known as Microsoft Foundation Classes and Template library(MFC And T). the MFC and T is made up of three parts. One of these is MFC library, which is corner stone of windows programming with VC++. The MFC is also referred to as an application framework because it provides a set of structural components that provide a readymade basis for almost in any windows program.

The Active Template Library (i.e. ATL) provides structures for writing specialized windows program. Finally object linking and embedding database template library(OLE DB) comes into play when working databases.

The important tools other than editor, linker, compiler of VC++ are AppWizard and class Wizard. They are not essential to the process of writing windows program, but provides such immense advantages in simplifying the development process, reducing the incidence of errors and shorting the time of computing the program.

1) Class Wizard:

The most powerful feature of VC++ is Wizard called Class Wizard. Class wizard writes code for you. Off course, Class Wizard is not a magic program. You have to tell it what you want to write for you. Suppose you need a new class , new virtual function as a new message handler function, class wizard writes the prototype to the function bodies and (if necessary the code to link the window)

message to the function in MFC support. Note that the class wizard neither recognizes nor deals with classes that are not based on MFC classes.

2) AppWizard:

AppWizard is a code generation that creates a working skeleton of the window application with features, class names and source code file names that you specify through dialog boxes. AppWizard code is smallest code, the functionality is inside the application framework based class. It's purpose is to get you started quickly with a new application.

Features Of VC++

- **Application Framework:**

  Like definition of application framework it is an integrated collection of object oriented software components that offers all that needed for generic application.

  An application framework is a super hit of a class library. An ordinary library is an isolated set of classes designed to be incorporated into any program but an application framework defines the structure of the program itself.

- **Documents And views:**

  The document view architecture is due case of the application framework. In simple terms the document view architecture separated data from the users view of data. One obvious benefit is multiple view of the same data.

The code which is written i.e. document class is responsible for sharing document (the program data). This code will enable you to save the program data into a file and lead data from previous saved files.
The code which is written in view class is responsible for what you see on screen.

Types Of Application :

a) Dialogue Based Application:

By selecting the dialogue based application,you were telling MFC AppWizard that you want to create a dialogue based program, which means that main window of the application is nothing but a dialogue box.

b) Single Document Interface (SDI):

As its name implies SDI application let you work with single document . in an SDI program you can work on only one document at any time. You cannot open several documents simultaneously. An example of SDI program is the notepad program.

c) Multiple Document Interface (MDI):

MDI program also enables you to work with documents. However an MDI program lets you work on an open several documents simultaneously. Microsoft Word is an example of MDI program.

System Design

Design is a meaning engineering representation of something that is to be built.

Software design is actually a multistep process that focuses on four distinct attributes of a program: data structure, software architecture, interface representation and procedural details. The design process translates requirements into a representation of the software that can be accessed for quality before coding begins.

At each stage, software design work products are reviewed for clarity, correctness, completeness and consistency with the requirements and with one another.

Designing the software means to plan how the various parts of the software are going to meet the desired goals. Various input and output screens have been designed using C++ graphics. According to the choice made by the user, the screen changes its appearance, displaying various new attributes.

FUNDAMENTALS OF DIGITAL IMAGEMORPHING

Digital image morphing compresses the high range of software, hardware & theoretical underpinnings.

The basic fundamental tasks required to perform an image morphing are as follows:

1) Image Acquisition
2) Image Storage
3) Image Processing
4) Display

1) Image Acquisition

The first step in image morphing is requiring on image, it can be done by on image sensor such as scanner. Thus two elements are required to acquire a digital image. The first is to sense the image by a physical device(e.g. scanner) that is sensitive to a band in electromagnetic spectrum such as x-ray, ultraviolet, visible band & that produces an electrical signal proportional to the level of energy sensed. The second is a digitizer, a device needed for converting electrical output of physical sensing device into digital form, for example, vidicon camera which we are using in our thesis.

Digitization of spatial co-ordinates (x, y) is called image sampling and amplitude digitization is called grey-level quantization.

2) Image Storage

Digital storage of for image morphing application falls into three principal categories:

Short term storage for use during morphing.

On line storage for fast recall.

Archival storage for infrequent access.

Storage is measured in bytes, kilobytes, megabytes, terabytes. One method for providing the short term storage is computer memory.

In on line storage, more recent technology, called magneto optical (MO) storage uses a laser and specialized material technologies. The key factor characterizing on-line storage is frequent access to data.

Archival storage is characterized by massive storage requirement but infrequent need for access. Magnetic tapes & optical disks are the usual media for archival application. The key problem with magnetic tape is a relatively short shelf life about seven years and need for a controlled storage environment.

Current write-once-read-many (WROM) optical disk technology can store on he order of 1 Gbytes on 5-in. disk. Unlike MO,WORM disk with larger form factor are available, with capability to store nearly 6 Gbytes on 12-in. disks & slightly more than 10 Gbytes on 14-in. disk WORM disks has got many advantages over magnetic tape such as they have a shelf life that exceeds 30 tears without special environment requirements.

WORM disks can also serve as on-line storage devices in application in which read-only operation are predominant. Thus, we care using a short-term storage and archival storage in our thesis.

3) Image Processing

Processing of digital images involves procedures that are usually expressed in the algorithmic form. Thus with the exception of image acquisition and display, most image processing functions can be implemented in the software. The only reason for specialized image processing hardware is the need for speed in some applications or to overcome some fundamentals computer limitations.

Thus today's image processing systems are a blend of specialized image processing hardware, with the overall operation being orchestrated by software running on host computer.

A significant amount of basic image processing software can now be obtained commercially. When combined with others software for applications such as spread sheets & graphics, provides an excellent starting point for the solution of specific image processing problems. Sophisticated display devices

and software for word processing and report generation facilitate presentation of results.

Image processing is characterized by specific solutions. Hence techniques that work well in one area can be totally inadequate in another. All that the availability of powerful hardware and basic software does is to provide a starting point much farther along than it was less than a decade ago.

4) Display

Monochrome and color TV monitors are principal display devices used in modern image processing systems. Monitors are driven by the outputs of a image display module in the backplane of the host computer or as a part of the hardware associated with an image processor. Other display media include random access cathode ray tubes (CRTs) and printing devices.

In random-access CRT systems the horizontal and vertical position of CRT's electron beam is controlled by a computer which provides two dimensional drive necessary to produce an output image and at each deflection point, intensity of beam Is modulated by using a voltage, that is proportional to the value of corresponding point in the numerical array, varying from zero intensity outputs for points whose numerical value corresponds to black, to maximum intensity for white points.

Printing image display are useful primarily for low resolution image processing works. One simple approach for generating grey-tone images directly on paper is to use the overstrike capability of a standard line printer. Other common means of recording an image directly on pare includes laser printers, heat sensitive paper devices and ink-spray systems. Thus we are using monochrome and colour T.V. monitor and printers in our thesis.

IMAGE FILE FORMAT

The image morphing software of this thesis performs the transformation of source image to destination image.

The very first step is obtaining an image. It is not a simple task, because usable image data in a standard usable format is needed.

The software in this thesis deals with tagged image file format. The tag image file format was designed to promote the interchange of digital image data.

Aldus (of page marker frame), Microsoft and several other computer and scanner companies decided to write an industry standard for digital image data communication. Their collaboration results in the TIFF specifications. It is natural for PC based processing, because most scanner manufacturers support the standard in their PC products.

The goals of the TIFF specifications are extensibility. TIFF must be extensible in the future. TIFF must be able to adopt the new types of images and data must be portable between different computer processors and operating systems.

TIFF must be revisable. IT is not read only format software. Systems should be able to edit, process and change TIFF files. The tag in TIFF refers to the files basic structure. A TIFF tag provides basic information about the image such as width, length and no. of pixels. Tags are organized in tag directories.

Tag directories have no set length or number, since pointers lead from one directory to another. The result is a flexible file format that can grow and service in the future.

The following are the existing standard tags given: -

S.N	TAG CODE	INFORMATION
01)	255	Sub file type
02)	256	image width
03)	257	image length
04)	273	strip offset

05)	277	samples per pixel
06)	258	bits per sample
07)	320	colour map
08)	278	rows per strip
09)	279	strip bytes count.

01) Sub file type

Tag = 255(FF)
Indicates the kind of data in the sub file.

02) Image width

Tag = 256(100)
The width (x or horizontal) of the image in the pixels.

03) Image length

Tag = 257(101)
The length (y or height or vertical) of image in pixels.

04) Bits per sample

Tag = 258(102)
The no. of bits per sample

05) Strip offset

Tag = 273(111)
The byte offset for each strip.

06) Strip byte counts

Tag = 279(117)
The no. of bytes in each strip.

07) Samples per pixels

Tag = 277(115)
The no. of samples per pixels.

08) Rows per strip

Tag = 278(116)

The no. of rows per strip.

IMAGE DATA BASICS

An image consists of two-dimensional array of numbers. The co lour or gray shade displayed for a given picture element (pixel) depends on the number stored in the array for that pixel.

The baseline TIFF specification supports four-image type. The type of an image is determined from tag no. 262

The four types are as follows: -

Bilevel image: -

A bilevel image is a one bit per pixel image that can be interpreted as either black on white (i.e. white = 0) or white on black (i.e. white = 1).

Gray scale images:-

This type image data is somewhat complex. In this each pixel takes on a value between zero and the number of gray scales or gray levels that scanner can record. These images appear like black and white photograph. They are black and white and shades of gray, while some 256 shades of gray. Since a person can distinguish about 40 shades of gray, a 16 shade image may appear rough, while a 256 shade image looks like a photograph.

Colour mapped image:-

Colour mapped images are also referred to as palette based images. The image type requires a palette containing color definitions. Pixels values are interpreted or indexed into palette. The image type is a natural choice for all 16 colour & 256 colour VGA modes.

Full colour RGB image:-

This type is most complex type. It is used for 15 bit and 24 bit colour images among others. In most cases, images of this type specify 8 bits per sample & 3 for samples per pixels.

We can obtain TIFF mages by using a desktop scanner & our own photograph colour photographs work well but black and white photos are better.

While scanning the biggest problem is file size. The scanner can 300 dots per inch (dpi) so, 3*5 inch photo at a 300 dpi provides 900*15000 pixels. At a eight bits per pixel (256 shades of gray). The image file comes to over 13,15,000 bytes. This is not practical for many applications so, select a one or two inch square in photo & work with it.

TIFF Specifications

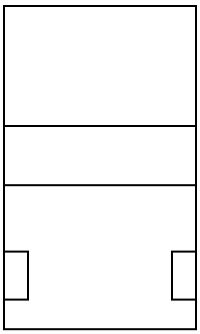
Table 1:-

Tag code	Information
225	Sub file type
256	Image width
257	Image length
273	Strip offset
277	Bits per pixel
258	Bits per sample
320	Colour map

Table 2:-

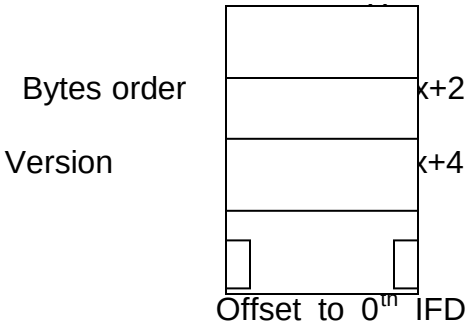
Type	Length of type
1=Byte	8 bit unsigned integer
2=ASCII	8bit bytes that store ASCII
3=Short	16 bit (2 byte) unsigned integer
4=Long	32 bit (4 byte) unsigned integer
5=Rational	Two longs : The first is numerator &the second is denominator

Header



offset

Directory entry



IFD:-

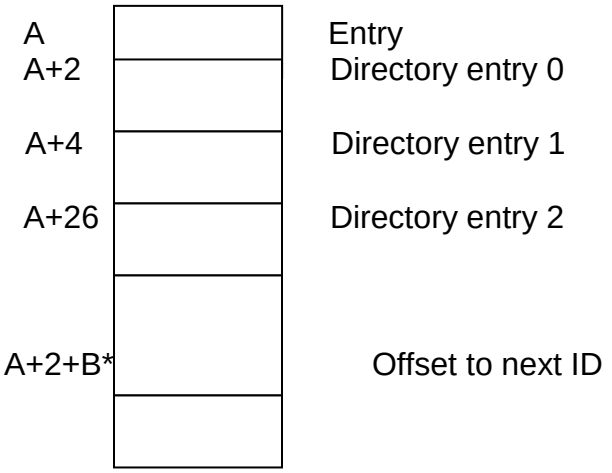


Fig a shows the structure of TIFF i.e. Tag Image file format. The first eight bytes are the header. These have same format on all TIFF files. These are the only items set in concrete for TIFF files. The remainder of the file differs from image to image. The image file directory i.e. IFD contains the number of directory entries and the directory entries themselves. The R.H.S. of the fig. shows the structure of each directory entry. Each directory entry contains a tag, that indicates the type of information the file holds, then it contains the data type of the information, the lengthy of the information, and a pointer to the information or the information itself.

Fig. b shows the beginning of TIFF file. The addresses are loaded on the left side in decimal and their bytes and their values are in the table in hex.

The first eight bytes are the header. Bytes zero and one tell whether the file stores a number with the most significant byte (MSB) first or least significant byte (LSB) first. If bytes zero and one are II (hex 4949) then the least significant byte is first, if the value is MM (hex 4d4d), then the most significant byte is the first. The example in fig. shows LSB first. Byte two and three give the tiff version, which should be 42(hex 2A) in all TIFF images. Bytes four to seven gives the offset to the first IFD. All offsets in TIFF indicate the location with respect to the beginning of the file. The offset in the fig. is 8, so the IFD begins in byte eight of the file.

THE IFD

The first two byte of the IFD gives the entry count. The content of the address eight is 27 (hex 1B). This indicates that the file has 27, twelve byte directory entries. The first two bytes of the entry contains the lag, which tells the type of information the entry contain. The directory entry 0 contains tag=255. This tag tells the sub file type as per table 1. The next two bytes of the entry give the data type of the information as per table 2. Directory entry 0 is type=3, i.e. a short which is two byte unsigned integer. The next four bytes of the entry give the length of the information. This length is not in bytes, but rather in multiples of the data types. If data type is short and length is one, then the length of the information is short i.e. two bytes. An entries final four bytes give either the value of the information or a pointer or a pointer to the value. If length of information is four bytes or less, then the information is stored here. If it is longer than four bytes, then a pointer to it is stored. The information in directory entry 0 is two bytes long and is stored here with a value 1.

As for the next two entries, the entry 1 has tag =256. This is the image width of the image in number of columns. The type is short and length of the value is one short or two bytes. The value 600 means that there are 600 columns in the image. The entry 2 has tag=257. This is the image length or height of image in number of rows. The type is short, the length is one short and the value is 602. Meaning that the image has 602 rows.

You continue through directory entries until you reach the offset to the next IFD. If this offset is 0, as in fig b, then no more IFD follows in the file.

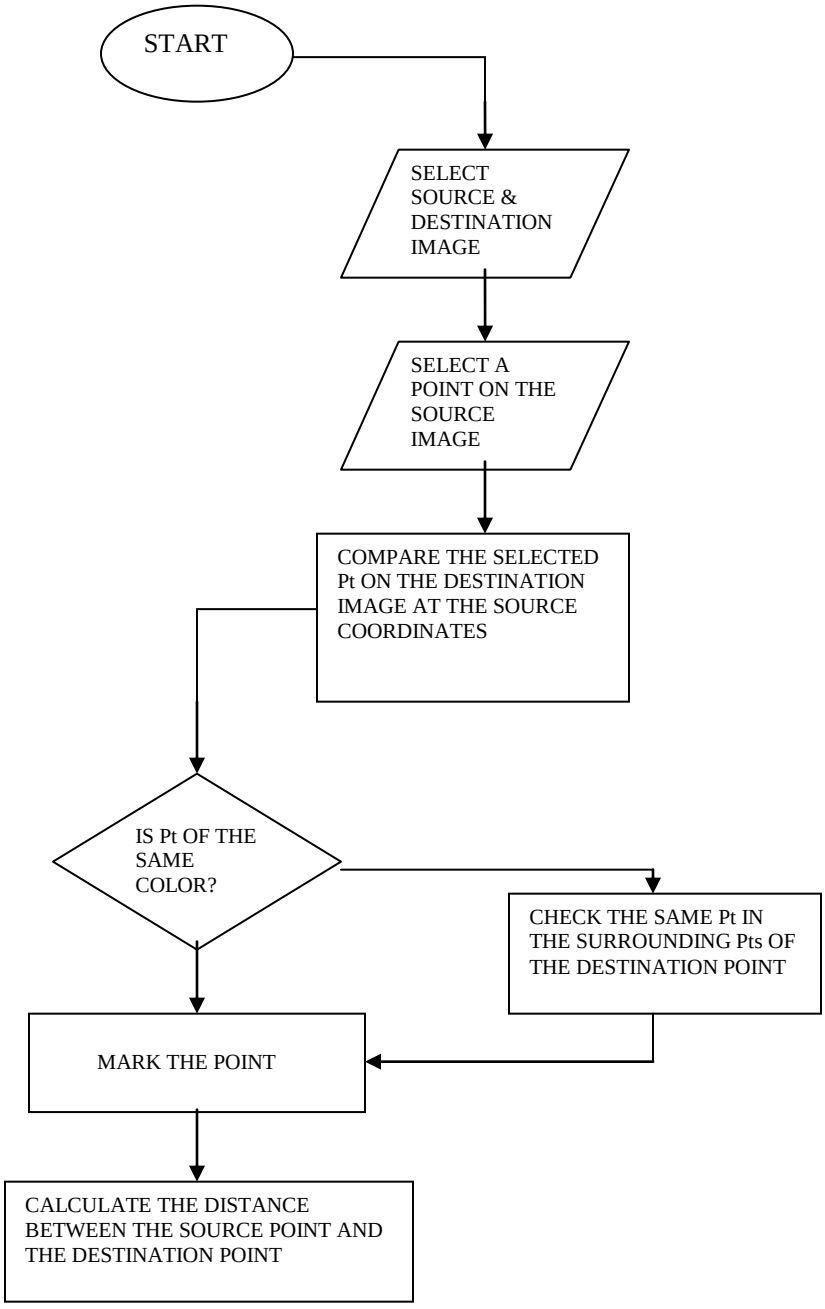
ADDRESS (decimal)	**CONTENTS** (hex)
Header	
0	49 49
2	2A 00
4	08 00 00 00
8	1B 00
0^{th} Directory entry	
10	FF 00 tag225
12	03 00 type=3(short)
14	01 00 00 00 length=1
18	01 00 00 00 value=1
1^{st} Directory entry	
22	00 01 tag=256
24	03 00 type=3
26	01 00 00 00 length=1
30	58 02 00 00 value=600
2^{nd} directory entry	
34	01 01 tag=257
36	03 00 type=3 (short)
38	01 00 00 00 length=1
42	5A 02 00 00 value=602

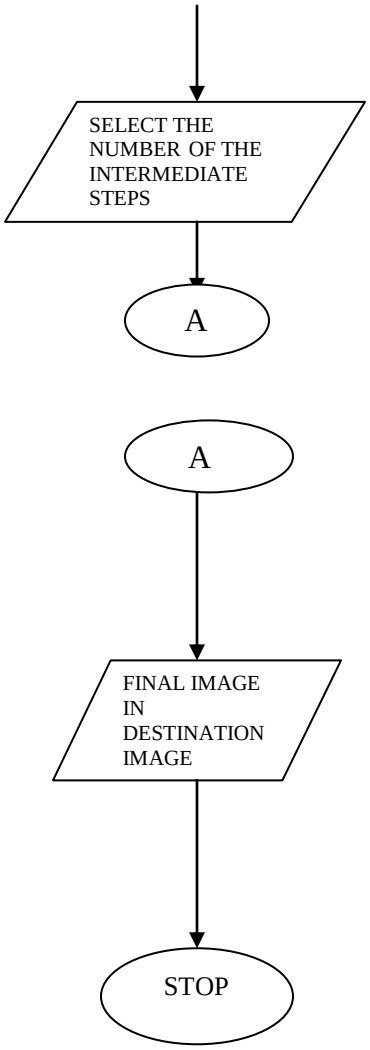
.
.

.

334 Offset to next IFD 00 00 00 00 (no more IFD)

FLOWCHART OF IMAGE MORPHING




SELECT THE NUMBER OF THE INTERMEDIATE STEPS
A
A
FINAL IMAGE IN DESTINATION IMAGE
STOP

Modular description of our thesis:

1. **MAP:-** In this function we are forming a particular square area. To do this we are passing starting point's co-ordinates and ending point's co-ordinates i.e. width and height of the square area. Also in this map function we are passing a fixed sized array. The colour which we present in the square shaped area will be mapped in the given array.

2. **FIND: -** In FIND function we are passing two structures and co-ordinates of the points as a parameter. Here the colour, which is present on the position, which we have passed as co-ordinates of a point in first structure, will get searched in the second structure at that point only.

3. **PLOT: -** With the help of this function actual morphing steps will take place. If user wants morphing effects in total number of the shades he had passed, he will get that effect displayed on the specified area of the screen by passing the total number of steps on line only.

In this way, we have used features of VC++ for our thesis – Image Morphing.

Testing Objective

Once the source code has been generated, software must be tested to uncover and correct as many errors as possible before delivery to the customer.

Testing is the process of executing a program, with the intent of finding an error.

If testing is conducted successfully, it will uncover errors in the software. It focuses on the logical internals of the software assuring that all the statements have been tested and ensured that the defined input will produce actual result that will agree with desired results.

In this thesis the testing methodology made use of is '**Black Box Testing**'.

Black Box Testing, also called as behavioral testing, focuses on the functional requirements of the software. That is, black box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

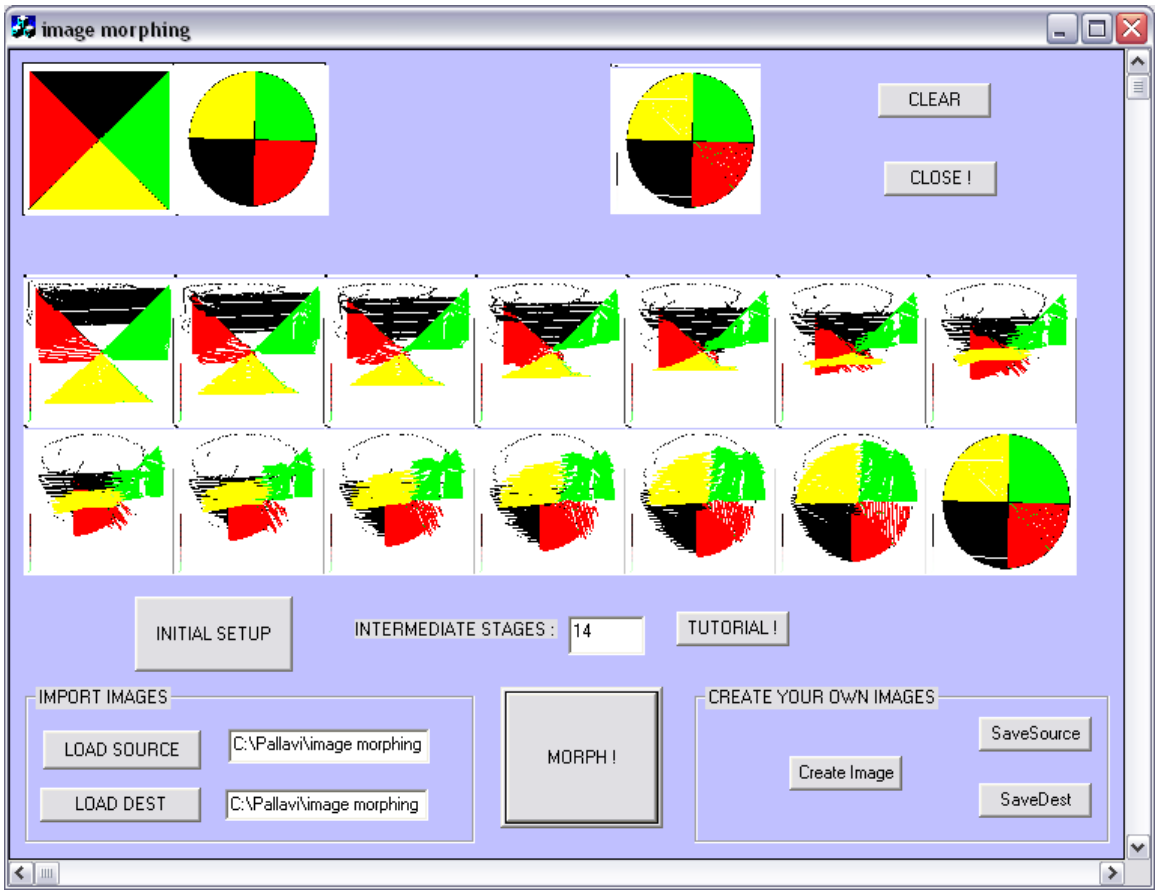
Black box testing attempts to find errors in the following categories:

- incorrect or missing function.
- interface errors.
- errors in data structures or external database access.
- behaviour or performance errors.
- initialization and termination errors.

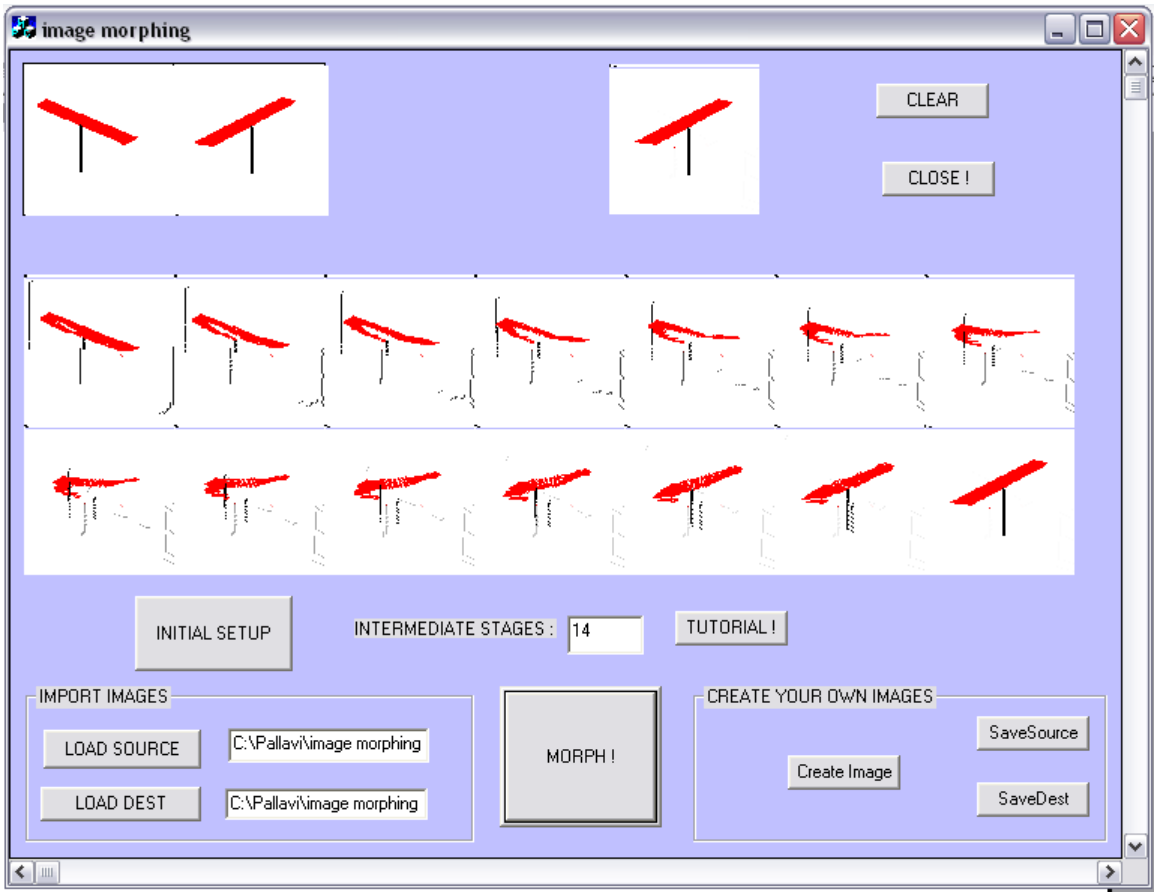
Limitation: The limitation is that the pictures need to be 16bit bmp to have good results.

Snapshots

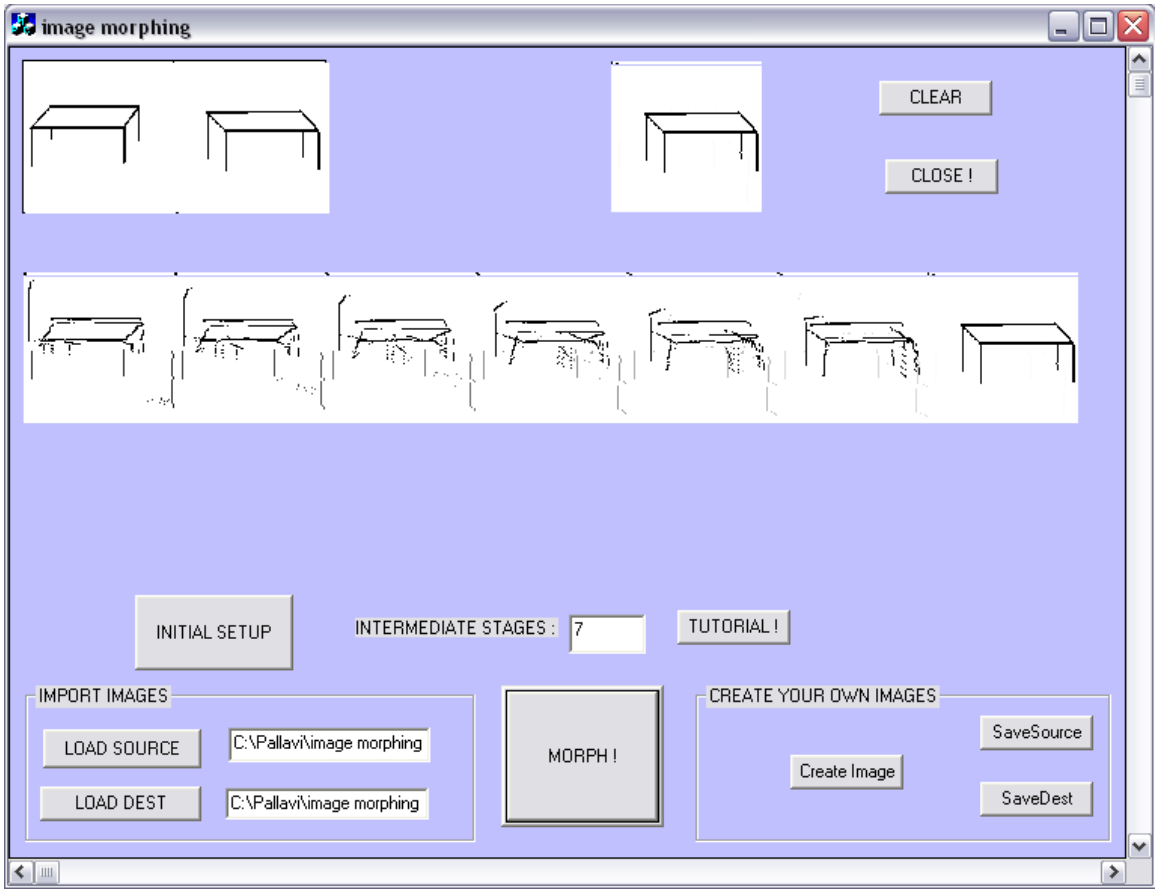
1. This is to show the effect of shape on morphing. Initial setup brings up two windows for source and destination images.The Source Image is selected by browsing to the appropriate location. The image comes up in the top-left window. Similarly the destination image is selected which also comes up in the top-left window next to the source image. The field for intermediate stages is selected (ideally to 14) to see the intermediate stages of the morphed image. The final intermediate image (along with the transforming stages) can be seen in the top center window. On clicking "MORPH!", the image gets morphed displaying the intermediate images and the final image at the top.



2. This is to show the rotation effect on morphing. Initial setup brings up two windows for source and destination images. The Source Image is selected by browsing to the appropriate location. The image comes up in the top-left window. Similarly the destination image is selected which also comes up in the top-left window next to the source image. The field for intermediate stages is selected (ideally to 14) to see the intermediate stages of the morphed image. The final intermediate image (along with the transforming stages) can be seen in the top center window. On clicking "MORPH!", the image gets morphed displaying the intermediate images and the final image at the top.



3. This is to show the effect of rotation of objects. Initial setup brings up two windows for source and destination images. The Source Image is selected by browsing to the appropriate location. The image comes up in the top-left window. Similarly the destination image is selected which also comes up in the top-left window next to the source image. The field for intermediate stages is selected (ideally to 14) to see the intermediate stages of the morphed image. The final intermediate image (along with the transforming stages) can be seen in the top center window. On clicking "MORPH!", the image gets morphed displaying the intermediate images and the final image at the top.



4.

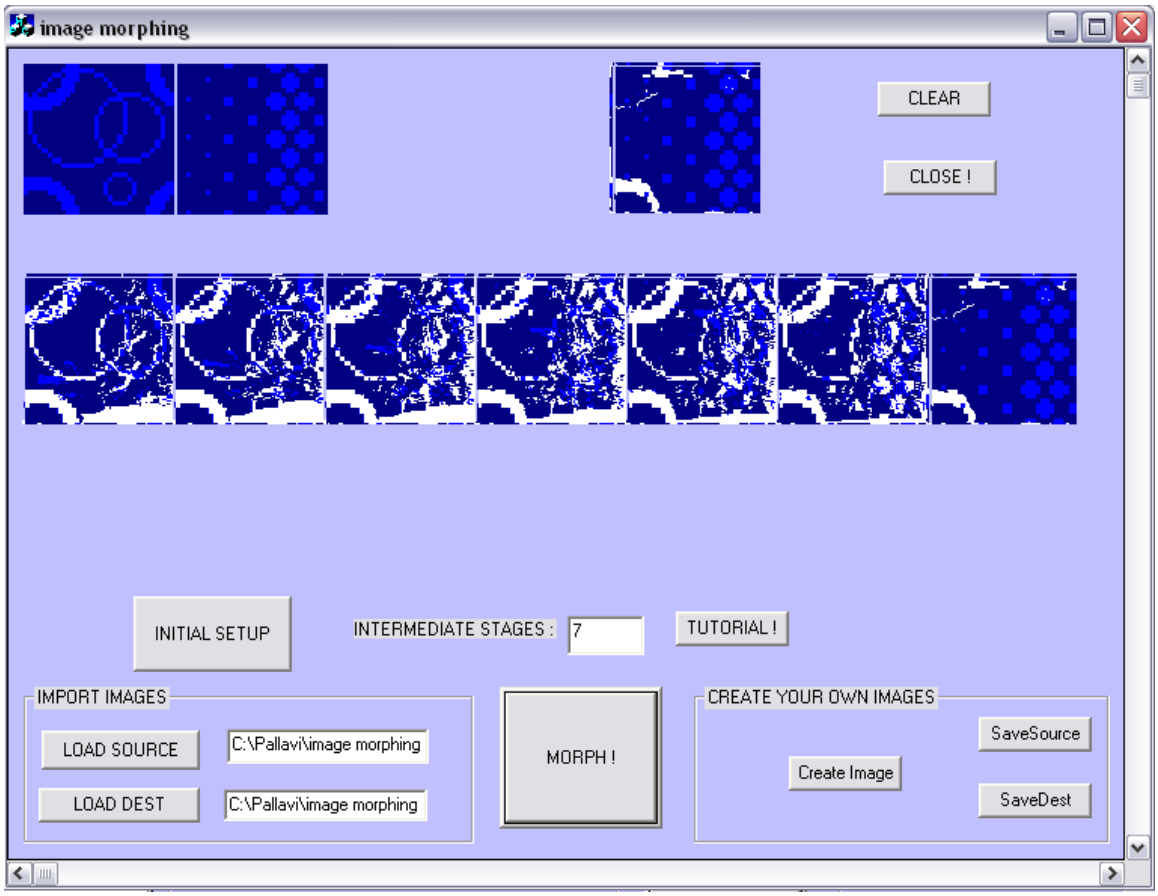


image morphing
CLEAR
CLOSE !
INITIAL SETUP
INTERMEDIATE STAGES :
7
TUTORIAL !
IMPORT IMAGES
LOAD SOURCE
C:\Pallavi\image morphing
LOAD DEST
C:\Pallavi\image morphing
MORPH !
CREATE YOUR OWN IMAGES
SaveSource
Create Image
SaveDest

5.

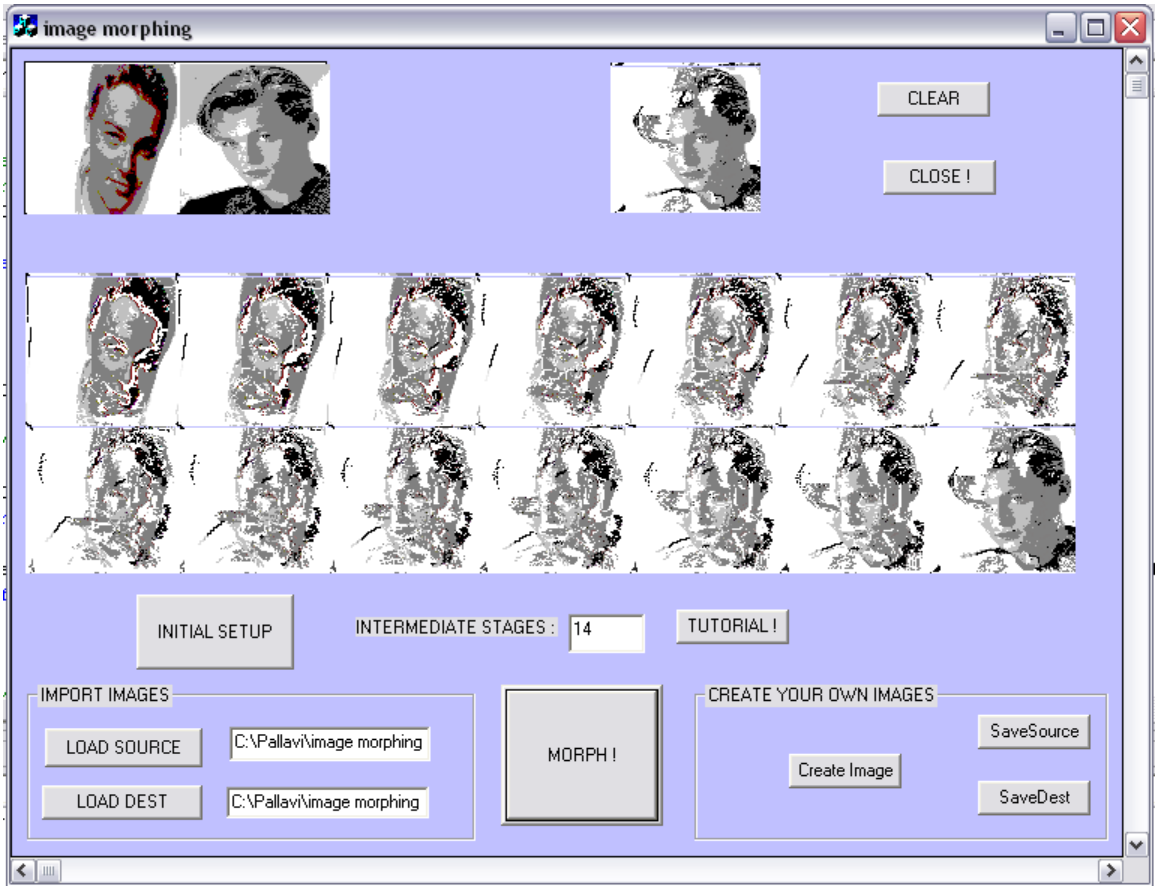


image morphing
CLEAR
CLOSE !
INITIAL SETUP
INTERMEDIATE STAGES :
14
TUTORIAL !
IMPORT IMAGES
LOAD SOURCE
C:\Pallavi\image morphing
LOAD DEST
C:\Pallavi\image morphing
MORPH !
CREATE YOUR OWN IMAGES
SaveSource
Create Image
SaveDest

6.


image morphing
CLEAR
CLOSE !
INITIAL SETUP
INTERMEDIATE STAGES :
14
TUTORIAL !
IMPORT IMAGES
LOAD SOURCE
C:\Pallavi\image morphing
LOAD DEST
C:\Pallavi\image morphing
MORPH !
CREATE YOUR OWN IMAGES
SaveSource
Create Image
SaveDest

APPLICATIONS:

The most famous usage (so far) and the once that started the morphing craze was in the movie Terminator 2 in which the futuristic assassin turns molecule metal, human being or whatever he work.

Using the morphing program, you can not only change a man into woman or a dog into a cat but you can also make an old person young again or a young person old.

Morphing programs can also be used to create time sequence effects. For instance, in a multimedia sequence, you could morph a small tree into a large tree to show effects of time. Most programs that allows you to morph also allow you to warp or bend images to create unusual or even frightening effects.

This technique has great use for commercial purpose like in advertisement and films for enhancing the effectively and also to give more realistic impression.

There is no doubt that all Doctors desire a quick convenient and accurate means of showing the patient of post-operative result. There are many existing ways to perform morphing but some systems have cumbersome, difficult and outdating morphing software. The Niamtu Imaging system has gone through grade length to provide user friendly, accurate and technologically advanced morphing application. Software will be automatically redraw your changes and the color, shadow and contrast will be calculated automatically. Most morphing applications take only seconds. Simply move the control points and the images is redraw in real time. Morphing can be done on any anatomical unit.

You may also zoom in one specific body part such as nose, chin, etc. and isolate the structure which give you mare control points and more precise changes without disturbing the rest of anatomy.

Besides morphing, the Niamtu Imaging system allows you to rotate any anatomy, whether you are straightening a tooth or rotating a nasal tip. This software makes it effortless.

SCOPE AND RESOLUTION

Our digital image morphing aims at the development of the basic image morphing system. This software can be used fro both gray scale i9mage and color mapped image i.e. most of the processing is also possible on both type of the image. There is no restriction to image size since this software is now been developed in visual C++ (i.e. one of GUI) is easy to handle for the new user. This software is hardware independent. Size of software is also considerably less. Software maintenance is very efficient.

Whatever we feel about image morphing we have achieved up to some extent. But this is not the end. Rather just the beginning and these are several interesting features such as morphing, image conversion etc.

While individual palette colors can be assigned any of the 16 million pellet colors, this does not guarantee that the physical display device is capable of displaying such as wide range of colors. This limitations is imposed by the graphics video card's limits much more than the video monitors most monitors are capable of near-infinite color resolution of course, a second limiting factor is also present. The human eye, even with excellent color vision, can only distinguish a fraction of the possible and the supported hues.

CONCLUSION

The Thesis has been designed and completed successfully. It is executing properly and is error free.

We have seen the image morphing right from introduction to application. Here we have used morph points instead of morph lines because of its simplicity, and also we have gone through the fundamentals of image morphing in which we have seen different processes used in image morphing.

I have also come to the conclusion that the Visual C++ programming language is one of the most stable programming languages and has a vast expanse and a very good scope for future software development.